Bachelorarbeit

Hochschule
Augsburg University of
Applied Sciences

Fakultät für
Informatik

Studienrichtung
Technische Informatik

Florian Pîrvu

Arbeitstitel: C++ Crossplatform Bibliothek zur Performanceanalyse von Anwedungen unter dynamisch generierten Lasten

Erstprüfer: Prof. Dr. Thomas Kirchmeier
Zweitprüfer: Prof. Dr. Hubert Högl

Abgabedatum: 20.01.2022

Hochschule für angewandte Wissenschaften Augsburg
An der Hochschule 1
D-86161 Augsburg
Telefon +49 821 55 86-0
Fax +49 821 55 86-3222
www.hs-augsburg.de
info(at)hs-augsburg-de

Fakultät für Informatik
Telefon +49 821 55 86-3450
Fax +49 821 55 86-3499

Verfasser der Bachelorarbeit
Florian Pîrvu
Asternweg 2
86399 Bobingen
Telefon +49 176 3693 7974
pflorian306@gmail.com

Inhaltsverzeichnis

1. Introduction

1.1 Testing

Software testing refers in the IT-Branch to a product with the main goal of finding bugs in a software program or application. These bugs refer to errors or faults and can occur because of a bad commands sequence specified in the program's source code.
A successful test can be achieved when its main requirements are met. Some examples would be the execution on different environments, time constraints or delivering expected outputs for randomly chosen inputs. Conditions are set by the tester and may vary depending on the use case.

1.2 Motivation

Most companies put a lot of effort in delivering high performance products to their customers. In order to do so, each of them test their gadgets, machines or software for possible failure scenarios. Now-a-days tests are being fully automated, which decreases the failure possibility that can happen because of human errors.[1]
Unfortunately the creation of fully automated tests is not as easy as it may sound. Many testing developers know the struggle of finding the right tools for the job and by the end of testing phase, their project is filled with unnecessary dependencies that will overload the program and occupy valuable memory.

"As ironic as it seems, the challenge of a tester is to test as little as possible. Test less, but test smarter"

Federico Teldo, Co-Founder Abstracta US

Additionally the problem enhances when a company reaches a certain size with a significant number of customers, which tend to run the product on different machines and architectures. In order to keep their customers, producers need to adapt their products to support newer or older machines. This makes software analysis even more difficult because of the increasing complexity, which comes with different systems. For this reason many companies dedicated themselves to creating programs that focus only on software examination and fixing. In time a new trend has been created with demands so high that rapidly developed itself into a new market section.

1.3 Growing markets

For the last decade the software testing market has been developing and now it has grown so big that it would be foolish to ignore it. According to ”Global Market Insights” the testing software market has grown up to 40 billion dollars by the end of 2020 and is predicted to grow up to 60 billion dollars in the next 6 years.[2]
”People may lie, but number don’t”. Multiple scientific papers and studies enforce this statement with different statistics of industries, which started adapting and reacting to this trend using the model ”EaaS ” which stands for ”Everything as a service” and created ”Testing as a Service”(TaaS). With this model customers not only pay for the current state of a product, but also for a service subscription, where they get updates and new features for the specific product and additional support from the company. The advantages that benefit the customer are set by the company for each of their subscription. (The basic rule is that you get more accurate results if you pay more). This service usually targets three groups: Developers, End Users and Certification Services.[3]

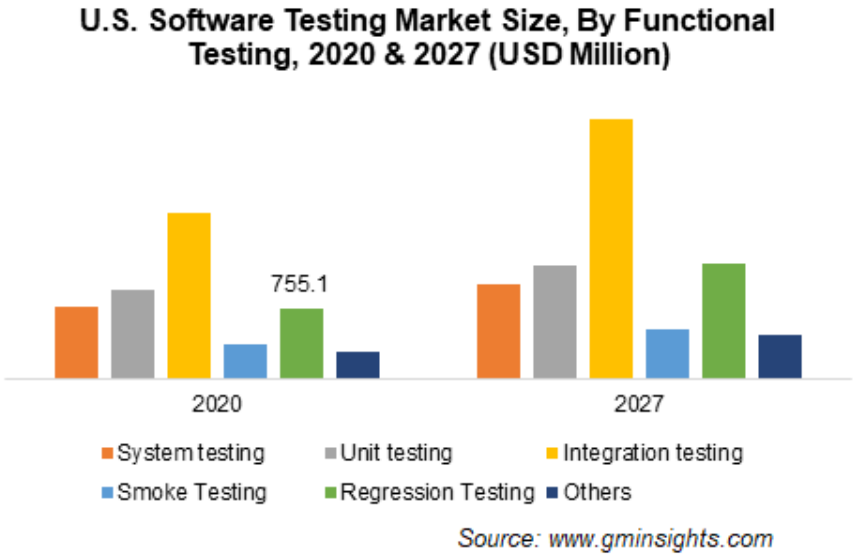


Abbildung 1.1: US Software Market Size[2]

1.3.1 Cost comparison

People often tend to overlook costs of testing tasks that are insignificant in comparison to the total price of the project. However if these costs are reoccuring, their total price can go up to 40% of the total project’s costs[4]. That amount of money covers not only for direct testing, which includes the staff, system and program testing, resources, computer time, etc. , but also for indirect testing, which refers to actions that take place because of poor direct testing, like rewriting code, additional analysis meetings or debugging.
It was proven that the cost of finding an error is about $50 on average [5], and is said that fixing an error after the software was released is four times more expensive than compared to if it was found during the testing phase.[6] To avoid these financial expenses, companies train their developers to consider failure scenarios of the product early in the development stage.

1.3.2 Common approaches

In order to deliver a defect-free product, managers create a testing strategy. To develop the most suitable strategy, they must identify the key components for it . This can be identified mostly by answering the following questions adapted from [6]:

1. Is the objective clear specified?
2. What tools will be used?
3. Is the system fully/partially automated?
4. How will the test benefit the project?

These questions should always be asked when a new feature that needs testing is being developed.

1.4 Current Situation

At the time of writing this (December 2021), there are four mostly used architectures in the IT-Branch. These are Windows, Linux and MacOS. Linux was developed with the UNIX system as its core, while MacOS is only based on UNIX, which means that these are similar but not entirely compatible with each other. The big difference comes with Windows.
Each operating system can deliver informations about its own computer.

For that each of them has its own unique program. Windows uses the ”Task-Manger” which comes with a GUI and shows real-time information about the CPU and memory of the computer, it comes with a list of processes and makes it possible for the user to manage them with only a couple of clicks.
Linux on the other hand is more text oriented. This operating systems comes with a built in command called ”top”. This also delivers informations about your system, but it doesn't allow you to manipulate processes like ”Task-manger” does. Although not very practicable for developers, these tools allow the user to measure performance in a normal state where your computer is not put under pressure, but the most interesting state is when the CPU needs to do a lot of operations and it needs to share its valuable time with other processes. To force such situations, the user can use online stress tests, which use the browser as an workload source, which for many is not very practicable, because it uses an additional program, which runs in the background. Online stress tests rely on internet to work and a stable ethernet or wifi connection to work. This way the results can be very inaccurate if the the network is under pressure.

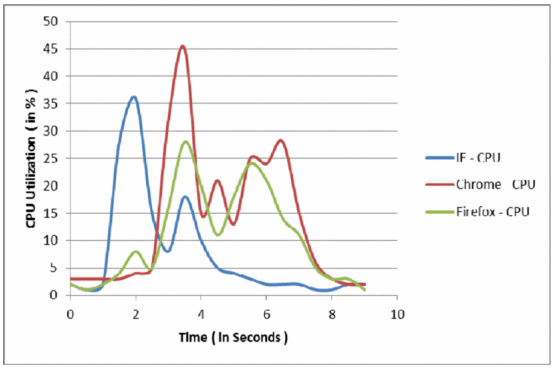


Abbildung 1.2: Browser CPU utilization comparison[7]

There are already some alternative libraries on the internet to recreate this method, but most of them are written in other programming languages. Adding them to a C++ Project would rise the complexity of the source code.

[0]Some companies also use ARM, but this is not an OS and per default doesn't come with any process control system. In order to do so one needs to write or use an extra library for that solely purpose

Some people would think ”Well C++ is a new and modern language. There must be something similar out there”. This statement is true and because of its young age there are a limited number of officially tested methods. You could also take a look at the headers implemented in the C++ standard library [8] to familiarize yourself with common algorithms and data structures.

1.5 Obective

The purpose of this thesis is the development, implementation and testing of a Library for C++ applications that creates an artificial workload and delivers system independent statistics based on multi-threading. This will allow developers to implement their own stress tests and run simulations with different workloads, process priorities and scheduling, which can be used in different manners to achieve real-time applications. These could be later automatized by using programs like Jenkins[9] or Gitlab[10].

1.6 Implementation

The library has two main components. The first one is the workload, which can vary depending of the user's input. This will create a system specific amount of threads and make them run a simulation function. The number of threads will stay constant and the time for running the workload task will be set accordingly for each input.
The second components is the system. This comes with functions, which can easily change a thread's priority, a system's scheduling policy and deliver statistics of the user's computer. Basic operations are implemented to work on most operating systems, but there are some exceptions because of the differences between OS implementations, which makes them independent from one another.

1.7 Advantages and Disadvantages

1.7.1 Pros

1. No additional expenses for third party software or subscriptions
2. Can be directly added to the source code, which makes testing easy
3. No external influence from the browser(ads) or internet connection, allowing efficient testing even offline
4. Allows the creation of tests based on user experience
5. Companies don't have to give their data

1.7.2 Cons

1. Tests don't come already prepared
2. Developers have add the source code to their build system or add the build files to their own if they already use CMake

1.7.3 Summary

With this library a company can save money, improve their product based on user feedback and create an internet independent test system for very minimal effort.

2. Systemgrundlagen

2.1 Prozesse

Ein Prozess ist ein Programm, welches sich in einem ausführbaren Zustand befindet. Er besteht aus zwei oder mehreren Befehle.
Bei der Erzeugung eines Prozesses bekommt dieser eine eindeutige Identifikationsnummer die sogenannte PID. Somit können zwei unterschiedliche Prozesse nicht die gleiche PID besitzen. Die Erzeugung neuer Prozesse erfolgt auf jedem Betriebssystem unterschiedlich. Wenn ein Prozess ein neuer Prozess erzeugt, dann spricht man von einem Kindprozess. Dieser besitzt eine PID und zusätzlich eine *Parent Process ID* kurz PPID, welche auf der PID des Elternprozesses zuweist.

2.2 Threads

Threads

2.3 Workload

The definition for workload varies from field to field. In the IT-branch it is defined as a unit of measure for your CPU (mostly in %). This tells the user how well his system can handle the number of current running processes. Most systems calculate their workload over a defined period of time. To understand this concept better, here is an example. Let's say a user starts an calculator program at time $t_0 = 0$. The application will finish initializing a GUI at time $t_{wait} = 2s$ and then wait for the user to enter some equation. After the equation was typed at $t_{input} = 5s$ and the Ënterkey was pressed, the app proceeds calculating and delivering the answer at $t_{done} = 6s$. So the CPU-time of the application is 3s(GUI initialization and calculation time). In order to get its workload the system has to divide this time by the time of the system needed for the whole process $\frac{t_{wait}+t_{calc}}{6s} * 100$. A detailed explanation on how to get these values will follow in Chapter (*Kapitelnr*).

3. Beispiele

Bla fasel...

Beispiele

3.1 Zitieren

Quellen[?, ?, ?, ?, ?] nicht vergessen. Dazu verwendet ihr bibtex.

3.2 Bild einfügen

3.2.1 Ein Bild skaliert

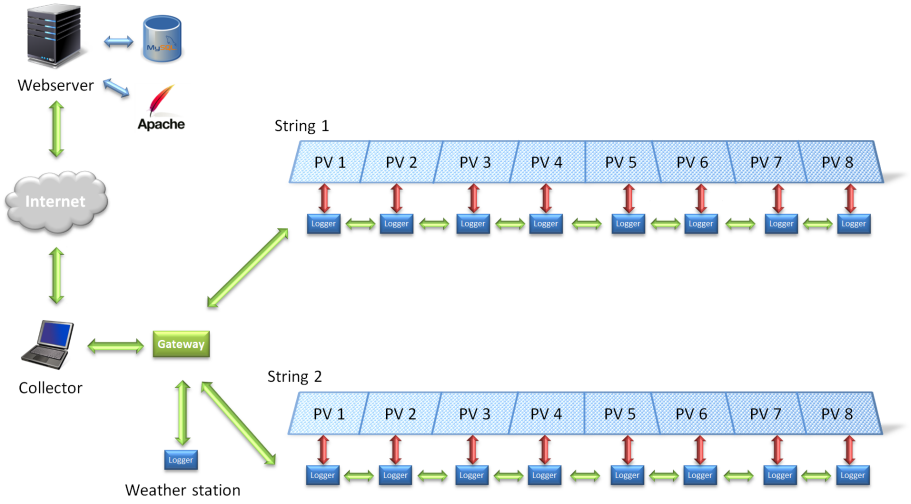
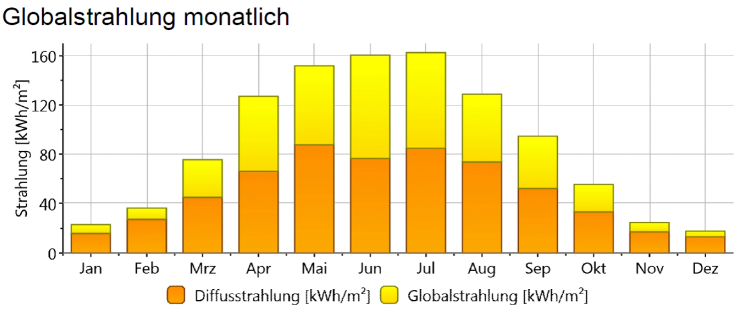


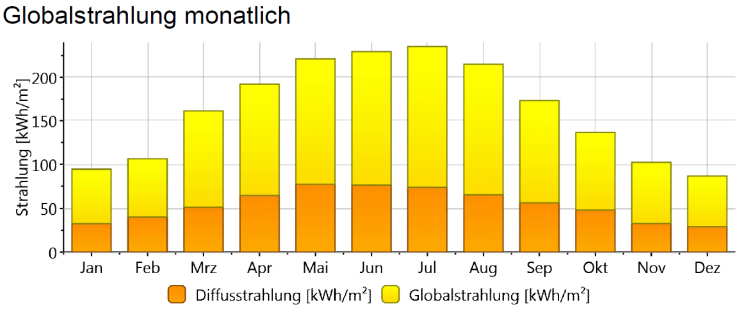
Abbildung 3.1: Beschriftungstext

3.2.2 Zwei Bilder nebeneinander oder untereinander

3.3 Tabellen



(a) Beschriftung Bild links



(b) Beschriftung Bild rechts

Abbildung 3.2: Beschriftung beide Bilder

Firma	Produkte / Lösungen	WEB
Concentrix (Soitec)	Module mit Konzentratoren (Fresnel-Linsen)	http://www.soitec.com
Isofoton	Module mit Konzentratoren (Fresnel-Linsen)	http://www.isofoton.com
Semprius	Module mit Konzentratoren (Fresnel-Linsen)	http://www.semprius.com
Azur Space	Mehrfach Junction Zellenhersteller	http://www.azurspace.com
Cyrium Technologies	Mehrfach Junction Zellenhersteller	http://www.cyriumtechnologies.com
Emcore	Mehrfach Junction Zellenhersteller	http://www.emcore.com

Tabelle 3.1: Hersteller von CPV-Produkten

Tabelle 3.2: Single-hop Scenario - Traffic Pattern

Pattern	Parameter	Distribution	Range/Values
Burst	Burst IAT	uniform	[9.9; 10.1] s
	Packets per Burst	constant	100
	Packet IAT	constant	0.02 s
	Packet Size	constant	1024 bit
	# Sources	-	2
	Offset	uniform	[0; 1] s
Single	Packet IAT	uniform	[0.9; 1.1] s
	Packet Size	constant	1024 bit
	# Sources	-	[10;20;30;40;50; 60;70;80;90;100]
	Offset	uniform	[0; 1] s

4. Analyse

Bla fasel...

4.1 Abschnitt 1

Bla fasel...

4.2 Abschnitt 2

Bla fasel...

4.2.1 Unterabschnitt

Bla fasel...

4.2.1.1 Unter-Unterabschnitt

Literaturverzeichnis

[1] Karuturi Sneha and Gowda M Malle. Research on software testing techniques and software automation testing tools. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 77–81, 2017.

[2] Shubhangi Yadav Preeti Wadhwani. Software Market Insights software testing market. https://www.gminsights.com/industry-analysis/software-testing-market, September 2021. Accessed: 2021-11-06.

[3] George Candea, Stefan Bucur, and Cristian Zamfir. Automated software testing as a service. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 155–160, New York, NY, USA, 2010. Association for Computing Machinery.

[4] Kevin J. Valle-Gómez, Pedro Delgado-Pérez, Inmaculada Medina-Bulo, and José Magallanes-Fernández. Software testing: Cost reduction in industry 4.0. In *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, pages 69–70, 2019.

[5] Edward F. Miller. Some statistics from the software testing service. *SIGSOFT Softw. Eng. Notes*, 4(1):8–11, January 1979.

[6] Vasundhara Bhatia, Abhishek Singhal, Abhay Bansal, and Neha Prabhakar. A review of software testing approaches in object-oriented and aspect-oriented systems. In M. N. Hoda, Naresh Chauhan, S. M. K. Quadri, and Praveen Ranjan Srivastava, editors, *Software Engineering*, pages 487–496, Singapore, 2019. Springer Singapore.

[7] Vishal Anand and Deepanker Saxena. Comparative study of modern web browsers based on their performance and evolution. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–5, 2013.

[8] Cpp Reference list of c++ standard library headers. https://en.cppreference.com/w/cpp/header. Accessed: 2021-11-06.

[9] Jenkins. https://www.jenkins.io/. Accessed: 2021-11-09.

[10] Gitlab. https://about.gitlab.com/. Accessed: 2021-11-09.

Abbildungsverzeichnis