

# Instrumentation and Performance Analysis of Distributed Systems with Freud

**Stefano Taillefert**

May 2021

*Supervised by*  
**Prof. Antonio Carzaniga**

BACHELOR PROJECT REPORT

# Abstract

Freud [1] is a software performance analysis tool that derives performance annotations from measurements of running systems. The goal of this project is to extend Freud to instrument and collect data from a distributed software system. This means augmenting the existing implementation to be able to link the data collected over distributed components using causal relations.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Performance analysis</b>	<b>3</b>
<b>3 Project design</b>	<b>4</b>
3.1 Freud . . . . .	4
3.2 Requirements and analysis . . . . .	4
<b>4 Implementation</b>	<b>5</b>
4.1 Technologies and tools used . . . . .	5
4.2 Issues . . . . .	5
<b>5 Evaluation</b>	<b>6</b>
<b>6 Conclusions</b>	<b>7</b>
6.1 Future work and possible developments . . . . .	7



## Chapter 1

# Introduction

Bla bla intro stuff

## Chapter 2

# Performance analysis

What is it, problem context, existing solutions in general

## Chapter 3

# Project design

### 3.1 Freud

Brief description, intro to Freud and PIN, what they do and how

### 3.2 Requirements and analysis

Goals, what I needed to implement, refer to the plan and list of tasks, what's the idea

Develop a simple distributed application based on an RPC library to be used as an initial test environment

Develop an instrumentation for the client side, server side, and – crucially – the RPC library

Devise a method to save and retrieve measurement logs from all the remote components

Design an algorithm to merge the logs from all the systems into a single coherent trace

Integrate said trace in the existing statistics tool (freud-statistics) to derive the performance annotations

Identify some third-party non-trivial distributed applications and analyze them with the created tool

Write the report, prepare the poster and presentation

Have a pizza

## Chapter 4

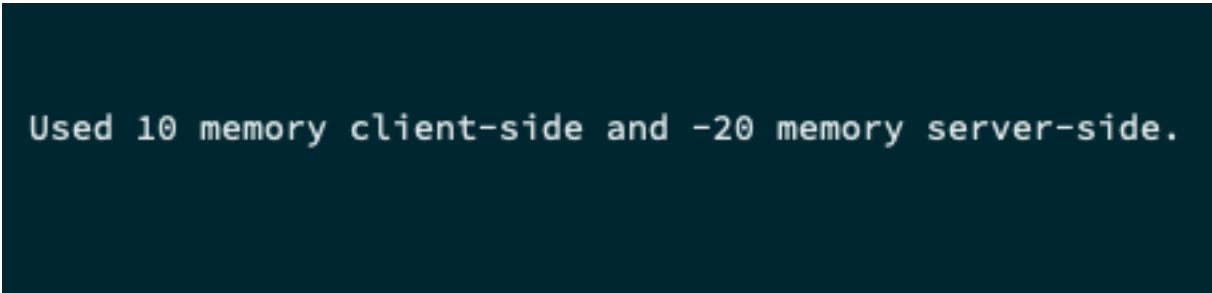
# Implementation

### 4.1 Technologies and tools used

Used gRPC [2] as the RPC library, freud-statistics [1] to analyze the output, bumped compiler to C++17 for `filesystem::exists()`

### 4.2 Issues

Made some choices to simplify structure initially, but then had to change it and refactor everything  
Problems making sense of the numbers returned by `freud-statistics` and unable to see what was in the binary file to check if I was dumping correctly  
Had to develop some particular encodings since passing from data to text and from text to data



```
Used 10 memory client-side and -20 memory server-side.
```

FIGURE 4.1: A very peculiar memory usage



## Chapter 5

# Evaluation

Code validation, analysis of results and practical applications, personal experience  
No coverage/automated tests due to the complexity and erratic nature of the software

## Chapter 6

# Conclusions

Results wrt objectives, limitations

Thanks to my advisor, Daniele and everyone that supported me

### 6.1 Future work and possible developments

Re: what the next guy will have to work on next year

Better handling of the data dumping/efficiency (separate thread)

Integration with PIN (very hard)

Support more complex features (re: freud's CLASS type or something)

# Bibliography

- [1] GitHub Inc. `usi-systems/freud`: Freud, a tool to create performance annotations for c/c++ programs. <https://github.com/usi-systems/freud>, 2021. Accessed on 15.05.2021.
- [2] gRPC Authors. Documentation | `grpc`. <https://grpc.io/docs/>, 2021. Accessed on 27.04.2021.