Eötvös Loránd University

Faculty of Informatics

Department of Software Technology

Log analyzer for real-time DSP scheduling framework

Supervisor:
Zoltán Gera
tanársegéd, MSc Computer Science

Author:
Hossameldin Abdin
Computer Science BSc

Budapest, 2021

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

Thesis Registration Form

Student's Data:

Student's Name: Abdin Hossameldin

Student's Neptun code: BELNRM

Course Data:

Student's Major: Computer Science BSc

I have an internal supervisor

Internal Supervisor's Name: *Zoltán Gera*

Supervisor's Home Institution: *ELTE IK*

Address of Supervisor's Home Institution: *1117 Bp. Pázmány Péter sétány 1/C*

Supervisor's Position and Degree: *tanársegéd, MSc Computer Science*

Thesis Title: Log analyzer for real-time DSP scheduling framework

Topic of the Thesis:

(Upon consulting with your supervisor, give a 150-300-word-long synopsis os your planned thesis.)

1 Introduction:

Logging various information regarding different aspects of projects is vital to measure the sanity and the behavior of a system. Unfortunately in many cases especially in a case of a huge amount of information to log, the advantage turns into an issue that takes time and effort from the developer(s) to be able to check the sanity of a created system or a program and from that point the logging becomes a burden which takes from the efficiency of the program without giving back the wanted/requested quality of results. Also the debugging in a real-time system is not possible, the log analyzing is the only option in such a system. From the issue described above, the idea of a log analyzer was born. The log analyzer will support the DSP (Digital Signal Processing) framework PipeRT which is developed at ELTE University.

2 General information about PipeRT:

PipeRT is a hybrid scheduling and data flow framework for DSP applications, which offers high performance and easy to use framework. (for more info: https://github.com/gerazo/pipert/blob/master/README.md)

3 The responsibility of the log analyzer:

The log analyzer should be a separate API which can communicate with the framework to have a low delay live sanity checking for the system, it should be able to represent the pipeline of the framework visually and it should spot the bottleneck in the system if any. The analyzer should generate statistics that can reflect the status of the system as a whole.

4 Goal:

Offering the developers of DSP applications who uses the PipeRT framework a detailed yet understandable representation of their development's pipeline. The analyzer is supporting the measurement oriented approach of the development.

Contents

Chapter 1

Introduction

1.1 Motivation

Logging is not an easy addition to any system but becomes useful only with a tool that knows how to extract valuable data from a huge stream and from these data can bring an overview, and statistics that describe the behavior and analyze it. The log analyzer became essential not only to support such a system and shows its flows, but also to visualize a picture to force the developer(s) to see what he/she never expected.

For all the reasons mentioned and more, building a log analyzer to show the bottlenecks and help the developer(s) of DSP (Digital signal processing) application who are using the PipeRT framework was a project eager to be born.

1.2 Thesis Structure

This thesis is composed of 4 main chapters, a bibliography, a list of figures, and a list of tables.

Chapter 2 is going to introduce the user documentation, including how to install and run the analyzer.

Chapter 3 contains the developer documentation with detailed Structure of the implementation, and its capabilities to be extended.

Chapter 4 is the conclusion, and the summary of the project can be found there, with ideas to be added to the project.

Chapter 2

User Documentation

This chapter contains a brief description of the project, a guide on how to install and run the analyzer, and the way to use.

2.1 Project Description

This project is a log analyzer working alongside the PipeRT framework with its profiler. The project aims to analyze the continuous stream of data coming from the PipeRT's profiler, in a server-client relationship.

The analyzer was build using python in the backends and Javascript in the frontend, it provides various checkers to investigate the sanity of the pipeline created by user using PipeRT, graphs associated with the measurements prepared by the checkers, and visualization of the pipeline and how the channels are structured.

The analyzer in itself was built to be an analyzing framework where extensibility was the key and will be in the design decisions, so as a result, it is relatively easy to add new features and already prepared to be extended by new checkers and measurements.

2.2 Installation Guide

PipeRT is currently supporting Linux only, but soon, it will support Windows as well. That said, the steps to install the analyzer are the same in both operations systems.

The `log_analyzer` folder inside the PipeRT project folder is where all of the Installation steps will take place.

Python 3.9 should be installed on the operating system. All the requirements can be installed by typing the following command in the terminal or the command prompt.

```
$ pip install requirements.txt
```

Code 2.1: Install requirements

It is also recommended to create a python virtual environment before installing the requirements, in order to have a separate environment for the analyzer. The commands to create and run the environment are:

```
$ python3 -m venv venv
$ source venv/bin/activate
```

Code 2.2: Create virtual environment

2.2.1 Running

In order to run the analyzer make sure to be inside the `log_analyzer` folder and type the following command:

```
$ python start.py
```

Code 2.3: Start application

This will start the application, so once you type `127.0.0.1:5000` in the browser (Firefox recommended). You will be able to see the following:

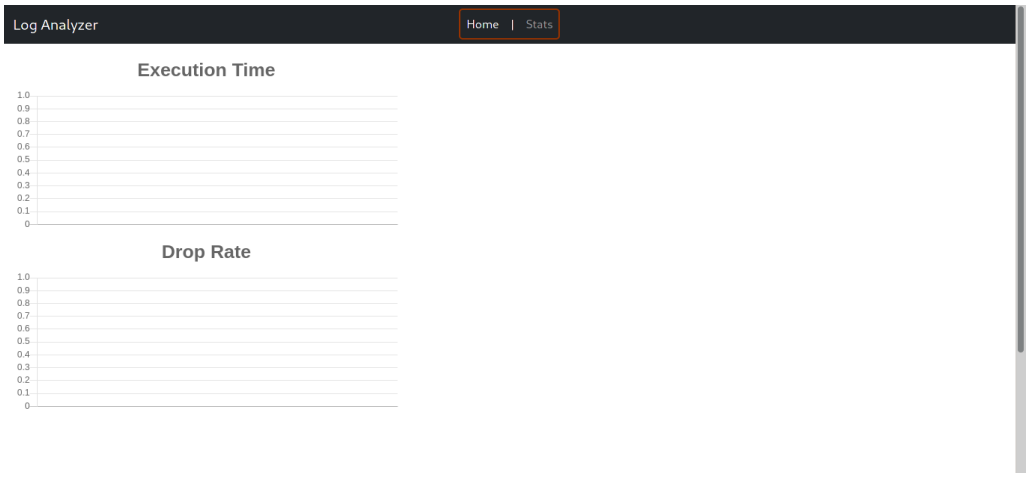

Figure 2.1: Start of the application

2.3 Client-Server Configuration

To establish a connection between the profiler (Client) and the log analyzer (server), there should be modifications in both sides.

2.3.1 Server Side

The profiler is the utility for monitoring the DSP pipeline and sending logs, it has 3 arguments, first is `destination_uri`, which describes the destination and the used protocol, udp and file are the protocol options in the profiler currently. The Second argument is `aggregation_time_msec` which is the time in milliseconds to wait before gathering monitoring data again, so it determines how often aggregated log data is sent to the log processor, if not given that means not to collect periodically. The third argument is `buffer_size`, it controls the size of buffer which is filled to be sent at once, the default value depends on the protocol chosen.

To establish a connection with the analyzer, the udp protocol is the one to choose, the IP and socket are based on the user preference.

Adding the profiler to the scheduler is the last step to configure the server-side, and the following example showing how to add the profiler.

```
pipert::Scheduler sch(0, pipert::Profiler("udp:127.0.0.1:8000"));
```

Code 2.4: Adding profiler

2.3.2 Client Side

On the client-side, the same port number that has been provided to the profiler should be added in the **config.json** inside the `log_analyzer` folder. same for the IP as well.

```
{
  "PORT": "8000"
  "IP": "127.0.0.1"
}
```

Code 2.5: connection configuration

An important note: The log analyzer will start analyzing and draw the visualization, at the end of the first packet cycle 2.4.1, so, the web page will be the same as 2.1 until the completion of the cycle.

2.4 Analyzer Configuration

The log analyzer is made to check the sanity and analyze various applications and systems, so having a configuration, was essential.

The configuration is a JSON (JavaScript Object Notation) file, the choice of the format to be JSON was due to its lightweight, well-known among developers, and simplicity. The config.json configuration file consists of 2 main parts, general and checkers' configurations. Let's enumerate these configurations.

```
{
  "PORT": "8000",
  "IP": "127.0.0.1",
  "PACKET_CYCLE_THRESHOLD": 1000,
  "checkers": [
    {
      "name": "frozen_checker",
      "enabled": true,
      "parameters": {
        "FROZEN_LIMITS": 20
      }
    }
  ]
}
```

Code 2.6: Sample configuration

2.4.1 General Configuration

Consist of 3 configurations, **PORT**, **IP**, and `PACKET_CYCLE_THRESHOLD`. The first two were discussed in 2.3.2.

The `PACKET_CYCLE_THRESHOLD` is an integer value that describes how many packets should the analyzer receive before running the checkers once and these packets'

events will be stored in the channels and when the cycle finishes (when the number of packets received is equal to `PACKET_CYCLE_THRESHOLD`), these events will be deleted from the channels and a new packets cycle starts.

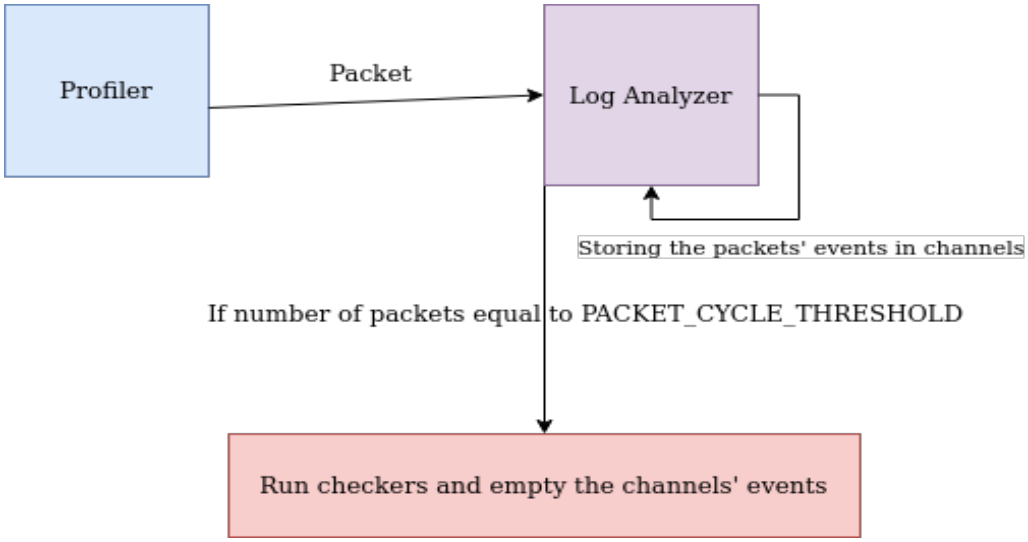

Figure 2.2: Packets cycle

2.4.2 Checkers' Configuration

As it is shown from 2.6, the checkers' configuration is a list of dictionaries where each dictionary is a checker's configuration. Each configuration contains the checker's name in the **name** field, whether it should work or not in the **enabled** field, and a **parameters** field where the parameter of the checker can be created or changed.

The **name** field should be the same name as the file of the checker without the extension (.py). And that is how the dynamic importing module in the project will be able to import the checker with its configuration **??**.

The **enabled** field is a boolean field to turn on or off the checker.

The **parameters** field is a dictionary with as many keys as the checker's parameters. These values can be changed based on the DSP application using the analyzer, and it is the user's responsibility to assign the appropriate values.

In the next section, we will discuss the implemented checkers in detail.

Phasellus tortor	**Aenean consequat**
Sed malesuada	Aliquam aliquam velit in convallis ultrices.
Purus sagittis	Quisque lobortis eros vitae urna lacinia euismod.
Pellentesque	Curabitur ac lacus pellentesque, eleifend sem ut, placerat enim. Ut auctor tempor odio ut dapibus.

Table 2.1: Maecenas tincidunt non justo quis accumsan

2.4.3 Sorok és oszlopok egyesítése

Mauris a dapibus lectus. Vestibulum commodo nibh ante, ut maximus magna eleifend vel. Integer vehicula elit non lacus lacinia, vitae porttitor dolor ultrices. Vivamus gravida faucibus efficitur. Ut non erat quis arcu vehicula lacinia. Nulla felis mauris, laoreet sed malesuada in, euismod et lacus. Aenean at finibus ipsum. Pellentesque dignissim elit sit amet lacus congue vulputate.

Quisque	**Suspendisse**		**Aliquam**		**Vivamus**	
	Proin	Nunc	Proin	Nunc	Proin	Nunc
Leo	2,80 MB	100%	232 KB	8,09%	248 KB	8,64%
Vel	9,60 MB	100%	564 KB	5,74%	292 KB	2,97%
Auge	78,2 MB	100%	52,3 MB	66,88%	3,22 MB	4,12%

Table 2.2: Vivamus ac arcu fringilla, fermentum neque sed, interdum erat. Mauris bibendum mauris vitae enim mollis, et eleifend turpis aliquet.

2.4.4 Több oldalra átnyúló táblázatok

Nunc porta placerat leo, sit amet porttitor dui porta molestie. Aliquam at fermentum mi. Maecenas vitae lorem at leo tincidunt volutpat at nec tortor. Vivamus semper lacus eu diam laoreet congue. Vivamus in ipsum risus. Nulla ullamcorper finibus mauris non aliquet. Vivamus elementum rhoncus ex ut porttitor.

Praesent aliquam mauris enim	
Suspendisse potenti	*Lorem ipsum dolor sit amet*
Praesent	Nulla ultrices et libero sit amet fringilla. Nunc scelerisque ante tempus sapien placerat convallis.
Luctus	Integer hendrerit erat massa, non hendrerit risus convallis at. Curabitur ultrices, justo in imperdiet condimentum, neque tortor luctus enim, luctus posuere massa erat vitae nibh.
Egestas	Duis fermentum feugiat augue in blandit. Mauris a tempor felis. Pellentesque ultricies tristique dignissim. Pellentesque aliquam semper tristique. Nam nec egestas dolor. Vestibulum id elit quis enim fringilla tempor eu a mauris. Aliquam vitae lacus tellus. Phasellus mauris lectus, aliquam id leo eget, auctor dapibus magna. Fusce lacinia felis ac elit luctus luctus.
Dignissim	Praesent aliquam mauris enim, vestibulum posuere massa facilisis in. Suspendisse potenti. Nam quam purus, rutrum eu augue ut, varius vehicula tellus. Fusce dui diam, aliquet sit amet eros at, sollicitudin facilisis quam. Phasellus tempor metus vel augue gravida pretium. Proin aliquam aliquam blandit. Nulla id tempus mi. Fusce in aliquam tortor.
Pellentesque	Donec felis nibh, imperdiet a arcu non, vehicula gravida nibh. Quisque interdum sapien eu massa commodo, ac elementum felis faucibus.
Molestie	Cras ullamcorper tellus et auctor ultricies. Maecenas tincidunt euismod lectus nec venenatis. Suspendisse potenti. Pellentesque pretium nunc ut euismod cursus. Nam venenatis condimentum quam. Curabitur suscipit efficitur aliquet. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Suspendisse potenti	*Lorem ipsum dolor sit amet*
Vivamus semper	In purus purus, faucibus eu libero vulputate, tristique sodales nunc. Nulla ut gravida dolor. Fusce vel pellentesque mi, vel efficitur eros. Nunc vitae elit tellus. Sed vestibulum auctor consequat.
Condimentum	Nulla scelerisque, leo et facilisis pretium, risus enim cursus turpis, eu suscipit ipsum ipsum in mauris. Praesent eget pulvinar ipsum, suscipit interdum nunc. Nam varius massa ut justo ullamcorper sollicitudin. Vivamus facilisis suscipit neque, eu fermentum risus. Ut at mi mauris.

Table 2.3: Praesent ullamcorper consequat tellus ut eleifend

Chapter 3

Fejlesztői dokumentáció

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis nibh leo, dapibus in elementum nec, aliquet id sem. Suspendisse potenti. Nullam sit amet consectetur nibh. Donec scelerisque varius turpis at tincidunt.

3.1 Tételek, definíciók, megjegyzések

Definition 1. *Mauris tristique sollicitudin ultrices. Etiam tristique quam sit amet metus dictum imperdiet. Nunc id lorem sed nisl pulvinar aliquet vitae quis arcu. Morbi iaculis eleifend porttitor.*

Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit, et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna. Phasellus faucibus varius purus, nec tristique enim porta vitae.

Theorem 1. *Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia. Etiam vel odio ante.*

Proof. Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui. □

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod, faucibus lectus sed, accumsan felis.

Remark. *Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec. Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus.*

Fusce in aliquet neque, in pretium sem. Donec tincidunt tellus id lectus pretium fringilla. Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris. Donec pretium et quam a cursus.

Note. *Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula.*

Ut sollicitudin tempus urna et mollis. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam. Donec eget tellus pharetra, semper neque eget, rutrum diam.

3.1.1 Egyenletek, matematika

Duis suscipit ipsum nec urna blandit, $2 + 2 = 4$ pellentesque vehicula quam fringilla. Vivamus euismod, lectus sit amet euismod viverra, dolor metus consequat sapien, ut hendrerit nisl nulla id nisi. Nam in leo eu quam sollicitudin semper a quis velit.

$$a^2 + b^2 = c^2$$

Phasellus mollis, elit sed convallis feugiat, dolor quam dapibus nibh, suscipit consectetur lacus risus quis sem. Vivamus scelerisque porta odio, vitae euismod dolor accumsan ut.

In mathematica, identitatem Euleri (equation est scriptor vti etiam notum) sit aequalitatem Equation 3.1:

$$e^{i \times \pi} + 1 = 0 \tag{3.1}$$

3.2 Forráskódok

Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor. Nam viverra orci id sapien sollicitudin, a aliquam lacus suscipit. Quisque ac tincidunt leo Code 3.1 and 3.2:

```
#include <stdio>

int main()
{
  int c;
  std::cout << "Hello World!" << std::endl;

  std::cout << "Press any key to exit." << std::endl;
  std::cin >> c;

  return 0;
}
```

Code 3.1: Hello World in C++

```
using System;
namespace HelloWorld
{
  class Hello
  {
    static void Main()
    {
      Console.WriteLine("Hello World!");

      Console.WriteLine("Press any key to exit.");
      Console.ReadKey();
    }
  }
}
```

Code 3.2: Hello World in C#

3.2.1 Algoritmusok

A general Interval Branch and Bound algorithm is shown in Algorithm 1. One of the following selection rules is applied in Step 3.

Példa forrása: Acta Cybernetica (ez egy link).

Algorithm 1 A general interval B&B algorithm

Funct IBB(S, f)

1: Set the working list $\mathcal{L}_W := \{S\}$ and the final list $\mathcal{L}_Q := \{\}$
2: **while** ($\mathcal{L}_W \neq \emptyset$) **do**
3: Select an interval X from $\mathcal{L}_W$ *Selection rule*
4: Compute $lbf(X)$ *Bounding rule*
5: **if** X cannot be eliminated **then** *Elimination rule*
6: Divide X into $X^j,\ j = 1, \ldots, p$, subintervals *Division rule*
7: **for** $j = 1, \ldots, p$ **do**
8: **if** X^j satisfies the termination criterion **then** *Termination rule*
9: Store X^j in $\mathcal{L}_W$
10: **else**
11: Store X^j in $\mathcal{L}_W$
12: **end if**
13: **end for**
14: **end if**
15: **end while**
16: **return** $\mathcal{L}_Q$

Chapter 4

Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

Appendix A

Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel

tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus, sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

List of Figures

List of Tables

List of Codes