

*Université du Québec à Chicoutimi*

*Département des sciences appliquées* **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Jean-Philippe Lapointe (LAPJ16078607)**

**Antoine Boucher (BOUA24089604)**

**Stéphane Tremblay (TRES13019002)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date de remise: 11 novembre 2020

**Rapport de laboratoire**

**Laboratoire 4**

**Architecture des logiciels (6GEI311)**

*Présenté à : Jeremy Bouchard*

**Table des matières**

[Introduction 1](#_Toc55999865)

[Méthode 1](#_Toc55999866)

[Boucle simple en langage interprété 1](#_Toc55999867)

[Boucle découpée en plusieurs processus en langage interprété 1](#_Toc55999868)

[Boucle simple en langage compilé 1](#_Toc55999869)

[Boucle découpée en plusieurs processus en langage compilé 1](#_Toc55999870)

[Résultats 2](#_Toc55999871)

[Discussion 2](#_Toc55999872)

[Conclusion 3](#_Toc55999873)

# Introduction

Le but de ce laboratoire est de comparer les performances d’un langage compilé, d’un langage interprété et de l’effet du multithreading/multiprocessus par l’utilisation d’une interface.

# Méthode

Le langage compilé est le c++ et le langage interprété est le python. L’interface est faite avec tkinter. Les grandeurs de temps sont calculées grâce aux horloges à haute précision du module time de python.

## Boucle simple en langage interprété

Cette boucle consiste simplement en la création d’un range de la grandeur du nombre d’itération puis de l’itération dans ce même range sans porter d’action grâce à la commande pass de python.

## Boucle découpée en plusieurs processus en langage interprété

Cette boucle utilise le module multiprocessing pour s’assurer d’une parallèlisation. La classe Pool est utilisé pour lancer plusieurs fois la boucle simple en langage interprété sur une fraction des itérations. Le nombre de processus lancé est de 4.

## Boucle simple en langage compilé

La boucle simple en langage compilé se résume en une seule fonction qui contient une boucle de type « for ». La boucle effectue l’incrément d’un compteur à chaque itération. La variable i qui agit comme itérateur est déclaré volatile pour éviter que le compilateur ne l’optimise et la retire.

## Boucle découpée en plusieurs processus en langage compilé

La boucle découpée en plusieurs processus commence par l’initialisation des threads. Un thread consiste en une fonction distincte qui reçoit en paramètre une partie du travail à effectuer soit effectuer une partie du compte dans le cas présent. Le thread reçoit une valeur de fin et effectue l’incrémentation du compteur avec ses données. Le nombre de thread est de 4

# Résultats

L’apparence de l’interface peut être vue ci-bas

Graphical user interface, text, application

Description automatically generated

Le tableau suivant présente les résultats de temps (sec) pour plusieurs nombres d’itérations et les différentes options.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Itérations-> | 1k | 1M | 10M | 100M | 1G |
| Compile thread | 0.0022 | 0.0025 | 0.010 | 0.055 | 0.36 |
| Compile simple | 0.000039 | 0.0058 | 0.027 | 0.19 | 1.75 |
| Interprété thread | 0.27 | 0.32 | 0.39 | 1.24 | 9.69 |
| Interprété simple | 0.00014 | 0.042 | 0.32 | 3.11 | 30.96 |

# Discussion

Deux observations intéressantes à faire.

Premièrement, les boucles en langages compilées sont toujours plus rapides que leurs équivalents en langages interprété pour un même nombre d’itérations, ce qui concorde avec la théorie.

Deuxièmement, les boucles sur plusieurs processus sont plus lentes que leurs équivalents simples pour les nombres d’itérations faibles deviennent plus rapides pour les nombres d’itérations important. Ceci est explicable par le fait que la gestion des threads demande un temps fixe et relativement important peu importe le langage.

# Conclusion

Pour conclure, ce laboratoire nous a permis d’observer les différences de vitesse entre les langages compilées et interprétés ainsi que les couts et bénéfices de l’utilisation de threads.