Vincent Desrosiers  
15039557

Projet Parallélisme

Travail Présenté à  
Gabriel Girard

IFT630 – Processus concurrents et parallélisme

Université de Sherbrooke  
Département Informatique  
13 avril 2017

Table des matières

[Introduction 3](#_Toc479543875)

[But du projet 3](#_Toc479543876)

[Fonctionnement du projet 3](#_Toc479543877)

[Spécifications 3](#_Toc479543878)

[Démarrage de l’application 4](#_Toc479543879)

[Initialisation du programme 4](#_Toc479543880)

[Reprise 4](#_Toc479543881)

[Nouvelle sauvegarde 4](#_Toc479543882)

[Suppression sauvegarde 4](#_Toc479543883)

[Exécution d’une sauvegarde 5](#_Toc479543884)

[Analyse dossier 5](#_Toc479543885)

[Exécution sur changement 5](#_Toc479543886)

[Conclusion 5](#_Toc479543887)

# Introduction

Ce document a pour but de décrire le déroulement du développement du projet de sauvegarde automatisé d’un répertoire en parallèle.

Ce rapport contient l’ensemble des éléments du projet de l’hypothèse jusqu’à la mise en pratique et analyse des résultats obtenus.

Du point de vue hypothétique et technique en premier lieu, il y aura la description du but du projet et ce que le projet tente d’accomplir. Par la suite, il y aura une description complète du fonctionnement de l’application permettant de comprendre ce que le programme fait et tente d’accomplir. Pour finir, il y aura une analyse des de l’avantage d’utiliser le parallélisme dans ce genre d’utilisation et les désavantages que ça peut apporter.

# But du projet

Le projet a pour premier but d’accélérer le traitement de copie de fichiers pour effectuer une sauvegarde d’un dossier vers un autre dossier ou un autre disque dur que le dossier source. Ce qui implique que le logiciel doit être capable d’effectuer une copie des fichiers d’un dossier selon les modifications apportés à un dossier. Autant pour l’ajout d’un fichier, de l’édition d’un fichier, la suppression de fichier et le changement de nom.

Ce projet a pour deuxième but, complémentairement au premier but, de paralléliser la copie de fichier pour éviter de faire une copie séquentielle de tous les fichiers d’un dossier à effectuer une sauvegarde. Ce qui a pour effet de faire une copie plus rapide selon le nombre de fichier et augmenter le nombre de copie selon un temps déterminé.

Ce projet a pour troisième et dernier but de paralléliser plusieurs sauvegardes séparément. Ce qui implique que tout le traitement de vérification de changement ainsi que l’exécution des opérations de fichier doivent se faire indépendamment de chacune des sauvegardes. Ce qui implique que chaque module de sauvegarde doit être indépendant du programme principal.

# Fonctionnement du projet

## Spécifications

Le programme est complètement fait en C# utilisant WPF pour l’interface graphique. Tous les processus en parallèle sont des threads attachés au programme principal pour garder un certain contrôle sur ceux-ci sans qu’ils bloquent et limite le bon fonctionnement de l’application principal.

La technologie utilisée pour faire la « Watch » (écoute) sur le dossier source d’une sauvegarde est l’outil [FileSystemWatcher](https://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher(v=vs.110).aspx)[[1]](#footnote-1) de Microsoft qui permet d’intercepter les messages du système d’exploitation en utilisant un appel API du système d’exploitation. Ce qui permet de faciliter toute la gestion de changement qui est moins lourd qu’un programme qui va chercher dans un intervalle la liste de tous les fichiers d’un répertoire donné.

## Démarrage de l’application

### Initialisation du programme

À l’ouverture du programme, celui-ci va tenter de charger le fichier de configuration de sauvegarde pour ouvrir et reprendre les sauvegardes déjà ajouté au programme. Par la suite, le programme alloue en mémoire chacune des configurations de sauvegarde et lance l’exécution de chacune des sauvegardes. Le lancement se fait en parallèle.

S’il n’y a pas de fichier de sauvegarde ou le fichier de sauvegarde est vide, le programme démarre normalement sans exécuter de routine de démarrage.

### Reprise

Lorsque le programme démarre, il y a une reprise qui est l’exécution d’une pré vérification du dossier source de chacune des sauvegardes chargées par le fichier de configuration pour reprendre la sauvegarde depuis que l’analyseur du dossier source était en pause ou que le programme principal était fermé. Cette reprise se fait complètement en parallèle pour chacune des sauvegardes. Cette fonctionnalité permet donc de transférer les fichiers qui ont été modifié dans le dossier source d’une sauvegarde. Ce qui inclut les fichiers ajoutés, modifié ou modifié. Le programme gère même les fichiers supprimé en faisant la vérification du dossier de destination pour voir les fichiers qui ne sont plus existant dans le dossier source.

## Nouvelle sauvegarde

Lorsqu’il y a une nouvelle sauvegarde qui est ajouté, le programme crée les entrés dans le fichier de configuration et lance la première exécution de la sauvegarde qui consiste à prendre la liste de tous les fichiers du dossier source et de les copier dans le fichier de destination. Chacun des fichiers sont envoyé dans un thread propre a lui-même pour effectuer sa copie. Par la suite, la copie démarre le watcher après avoir lancé chaque copie.

## Suppression sauvegarde

La suppression de sauvegarde permet simplement de supprimer la sauvegarde du fichier de configuration. Cette étape permet aussi de désactiver normalement le watcher et arrêter le thread de traitement du watcher.

## Exécution d’une sauvegarde

### Analyse dossier

Pour chaque sauvegarde, le watcher écoute les changements lancés par le système d’Exploitation et exécute des actions en conséquence des changements obtenus. Principalement pour ce projet, le watcher écoute pour tout changements à un fichier à l’exception des sous répertoires du dossier source pour ajouter une couche de simplicité et de diminution de chance d’erreurs dans le cadre de ce projet.

### Exécution sur changement

Pour chaque changement que le watcher reçois, il lance un nouveau thread sur la fonction du type de changement pour effectuer les traitements nécessaires à une certaine modification du contenu du répertoire source. Chaque thread se termine lorsque le traitement est terminé ou lorsque le nombre d’essai atteint son maximum.

# Conclusion

Tout au long du développement du projet, certains bons points et certains mauvais points de l’utilisation des threads entre en compte.

Une grande remarque est que la copie est parallèle de plusieurs fichiers n’est pas optimal surtout lorsque le répertoire source est dans le même disque que le répertoire de destination. Le transfert est quand même plus rapide lorsqu’il y a plusieurs petits fichiers et un gros fichier puisque l’utilisation des threads permet de transférer les petits fichiers sans attendre que le gros fichier soit terminé. De plus, l’utilisation des threads excessif pour chacun des fichiers est quand même lourd à gérer et peut prendre beaucoup plus de temps lorsque les fichiers sont assez gros.

Pour les bon points, l’utilisation du parallélisme permet de faire une gestion en temps réel de la modification de fichier dans plusieurs répertoire source. Ce qui permet donc d’éviter d’utiliser des attentes actives ou des vérifications synchronisées en temps pour détecter les changements en plus de faire une séquence dans l’ordre de vérification de chacune des sauvegardes.

Pour jumeler les bons points et les mauvais points, cette application de sauvegarde est tout de même utile puisque même si le traitement parallèle de plusieurs gros fichiers n’est pas optimal, le programme permet tout de même de faire une sauvegarde la plus complète possible en optimisant le nombre de fichiers sauvegardé dans un espace de temps.

1. Source https://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher(v=vs.110).aspx [↑](#footnote-ref-1)