UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO APLICADA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

USO DE PARALELISMO PARA APRIMORAR O SOFTWARE DE BUSCA POR INDÍCIOS DE PLÁGIO MISS MARPLE

MONOGRAFIA

Ricardo Bianchim Gomes

Santa Maria, RS, Brasil

2015

USO DE PARALELISMO PARA APRIMORAR O SOFTWARE DE BUSCA POR INDÍCIOS DE PLÁGIO MISS MARPLE

Ricardo Bianchim Gomes

Trabalho de Graduação apresentado ao curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Prof.ª Dr.ª Roseclea Duarte Medina

Trabalho de Graduação Nº 410

Santa Maria, RS, Brasil

2015

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Bacharelado em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

USO DE PARALELISMO PARA APRIMORAR O SOFTWARE DE BUSCA POR INDÍCIOS DE PLÁGIO MISS MARPLE

elaborado por
Ricardo Bianchim Gomes

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Roseclea Duarte Medina, Prof.ª Dr.ª
(Presidente/Orientador)

Carlos Raniery, Prof. Dr. (UFSM)

Raul Ceretta Nunes, Prof. Dr. (UFSM)

Santa Maria, 11 de Dezembro de 2015

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

USO DE PARALELISMO PARA APRIMORAR O SOFTWARE DE BUSCA POR INDÍCIOS DE PLÁGIO MISS MARPLE

Autor: Ricardo Bianchim Gomes
Orientador: Prof.ª Dr.ª Roseclea Duarte Medina
Data e Local de Defesa: Santa Maria 11/12/2015

O trabalho desenvolvido teve por objetivo o aprimoramento do *software* de busca por indícios de plágio *Miss Marple* através da aplicação de técnicas de processamento paralelo e distribuído. Inicialmente, foi identificado que esta ferramenta, em sua versão original, apresentava um longo tempo de processamento de documentos submetidos, limitando sua aplicabilidade a poucos documentos.

Os testes preliminares demonstraram baixo aproveitamento de arquiteturas multi-*core* dos processadores atuais, apontando a necessidade de desenvolvimento de uma nova versão da ferramenta, denominada *Parallel Miss Marple*. Esta nova versão da ferramenta é capaz de tirar proveito de arquiteturas de computação paralela e distribuída. Para obter aproveitamento das arquiteturas multi-CPU, faz-se uso de programação em *threads*. Já para distribuição em multicomputador, é utilizado a API Java RMI.

A nova ferramenta foi testada através da submissão de um conjunto de textos digitais, onde foi aferido o tempo de processamento das versões original e paralelizada do aplicativo, através do uso de uma ferramenta *profiler*. Os testes apontam ganho satisfatório de desempenho através da utilização da nova versão do programa.

Palavras-chave: Paralelo. Plágio. *Miss Marple*. *Multi-core*. Distribuído

ABSTRACT

Undergraduate Final Coursework
Computer Science
Federal University of Santa Maria

PARALELISM USAGE FOR IMPROVING THE MISS MARPLE PLAGIARISM CHECKING TOOL

Author: Ricardo Bianchim Gomes
Advisor: Roseclea Duarte Medina
Defense Place and Date: Santa Maria, December 11, 2015

The developed work had as objective the improvement of Miss Marple plagiarism checking tool, by using parallel-distributed processing techniques. Firstly, it was identified that tool, in its original version, used to take long time for processing documents. This fact limits its usage and applicability to a small set of documents.

Preliminary tests had shown that tool takes few advantage of current multi-core architectures, suggesting the need of developing a new version of the tool, so named *Parallel Miss Marple*. This new version of the tool is able of better extracting the power of current parallel and distributed computing architectures. For taking advantage of multi-CPU architectures, the new tool makes use of thread programming paradigm. For multicomputer distribution, it is used the Java RMI API.

The new tool has been tested by submitting a set of digital texts documents, where the processing times of the original and the parallelized versions of the application where measured by using a Profiler tool. Tests show a satisfactory increase in performance by using the new version of the program.

Keywords: Parallel. Plagiarism. Miss Marple. Multi-core. Distributed.

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface*

CPU - *Central Processing Unit* - Unidade de Processamento Central

DIP - Detector de Indícios de Plágio

DOC - Documento do Microsoft Word.

DOCX - Documento do Microsoft Word, baseado em XML

GPU - *Graphics Processing Unit* - Unidade de Processamento de Gráficos

IDE - *Integrated Development Environment*

NFS - Network File System - Sistema de Arquivos em Rede

RMI - *Remote Method Invocation* - Invocação de Método Remoto

SUMÁRIO

1 INTRODUÇÃO

O plágio é uma prática muito frequente, principalmente em se tratando de trabalhos no meio acadêmico. Ele pode ser definido como "ato de assinar como seu, trabalho ou ideia de outrem.", segundo o dicionário Aurélio. As causas são as mais diversas, estando entre elas, o acesso facilitado a uma grande quantidade de informações que é proporcionado pela Internet, o desconhecimento por parte dos alunos, a falta de orientação sobre o que caracteriza o plágio, e também, a desonestidade intelectual (ARENHARDT, 2013).

Ainda segundo Arenhardt (2013), "a Internet é a fonte de pesquisa mais utilizada". Pode-se citar a facilidade na obtenção em resultados de pesquisa, proporcionada pelos motores de busca como uma justificativa para este fato. Esta grande fonte de informações aliada à falta de instrução, inexperiência em escrita de textos, ou ainda, a má índole, acaba por se tornar uma ferramenta que potencializa a questão do plágio, ideia que é sustentada por Moraes (2004).

Baseado no fato apresentado no parágrafo anterior, a análise de plágio em trabalhos vem se tornando uma tarefa cada vez mais complexa para os avaliadores. Assim sendo, muitos *softwares* passaram a ser desenvolvidos com o intuito de executar análises em textos digitais, facilitando a tarefa de detecção de indícios de plágio. Conforme também é afirmado por Arenhardt (2012), é importante contar com uma ferramenta computacional que auxilie no processo de busca por indícios de plágio.

Um exemplo destes *softwares* é o *software* livre Miss Marple (ARENHARDT, 2013), desenvolvido em um projeto de mestrado do Programa de Pós Graduação em Informática (PPGI) da Universidade Federal de Santa Maria (UFSM). Durante o uso daquela ferramenta, pode-se observar boa precisão nos resultados do processo de análise de documentos. Também é possível verificar um bom tempo de processamento dos mesmos[1], no caso da submissão de *um único* arquivo para análise.

Porém, nem sempre o usuário responsável pela revisão de trabalhos efetuará análise em um único arquivo. Um professor, por exemplo, poderia examinar trabalhos de uma turma inteira. E neste caso, ocorre que os documentos são

[1] Comparado com *softwares* similares.

alocados em uma fila e em seguida são verificados somente um por vez, ou seja, sequencialmente. No cenário escolar e acadêmico, onde turmas frequentemente possuem trinta alunos ou mais, a análise sequencial de documentos pode se demonstrar algo inviável para o revisor, devido ao excesso de tempo consumido por essa forma de processamento.

1.1 Motivação

Considerando o exposto, é muito provável que o computador do usuário acabe ficando subutilizado, já que os computadores atuais são providos de dois ou mais núcleos de processamento. Já são comuns máquinas com processadores de quatro, oito ou mais núcleos. Portanto, é provável que a análise sequencial de um lote de documentos acabe demorando mais tempo que seria necessário por meio do uso de alguma forma de divisão de trabalho.

Com base nisto, surge a motivação para a realização deste trabalho. Ao se submeter diversos arquivos para análise, existe a necessidade de alguma forma de otimização para o processo como um todo, a fim de que o usuário da aplicação tenha uma melhor experiência de uso. Ou ainda, a fim de que o sistema seja mais adequado ao uso de grandes lotes de documentos.

A escolha do *software* Miss Marple em específico se deu devido a este se tratar de um trabalho já previamente desenvolvido no Grupo de Redes e Computação Aplicada (GRECA). Outro fator decisivo para a escolha é o fato do aplicativo demonstrar boa precisão de resultados de análise comparado a aplicativos similares, conforme é afirmado por Arenhardt (2013) .

1.2 Objetivos

O objetivo deste trabalho é aprimorar o *software* de análise de plágio Miss Marple, através da utilização do paralelismo. Pretende-se mitigar a questão do tempo de análise de grandes documentos de texto digitais ou grandes lotes de documentos. Para tal, serão explorados os recursos de programação paralela através do uso de *threads*, assim como recursos de computação distribuída.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são abordados termos e conceitos importantes para o desenvolvimento do trabalho. São apresentados o funcionamento do programa Miss Marple em sua versão original e alguns conceitos, como a caracterização do plágio, *stemming*, *stopwords*, paralelismo. Ainda, são reportados os trabalhos relacionados ao mesmo assunto, encontrados na bibliografia.

2.1 Caracterização do Plágio

O plágio pode ser definido como “assinar ou apresentar como seu (obra artística ou científica de outrem). Imitar (trabalho alheio)”, conforme definição encontrada no dicionário Aurélio. Na literatura, existem classificações do plágio quanto a sua natureza. Kirkpatrick (2001) classifica essa prática em quatro subtipos, sendo eles:

- Plágio direto: cópia literal, palavra por palavra, sem indicar a fonte onde se obteve tal material.
- Referência Vaga ou Incorreta: o escritor não especifica com clareza quais trechos são da autoria de terceiros.
- Tomar Emprestado de Outras Pessoas: consiste em apresentar um trabalho que foi escrito por outra pessoa. Pode ser considerado também, um caso específico do plágio direto.
- Plágio Mosaico: trata-se de uma cópia indireta, onde o escritor copia trechos de um texto, alterando poucas palavras, sem citar o autor original.

2.2 *Stemming*

Stemming é o nome dado a uma metodologia de extração do radical (*stem*) de diferentes variantes de uma palavra, muito utilizada na área de processamento de textos para recuperação de informações (ORENGO *et al.*, 2001). Por exemplo, tem-se o seguinte: as palavras "estudando", "estudo", "estudado", "estudar", possuem o mesmo *stem* "estud". O *stem* não necessariamente é uma palavra válida da língua em que o texto está escrito, mas sim um tronco de derivação para um grupo de palavras similares em significado.

Segundo Arenhardt (2013), os motores de busca utilizados durante a busca por indícios de plágio não fazem a distinção entre palavras diferentes com o mesmo radical, tratando-as como sinônimos. Isto pode levar a uma pesquisa incorreta, trazendo resultados que não refletem o conteúdo que está sendo analisado no documento.

Porém, com a correta aplicação de *stemming*, esse problema é mitigado, uma vez que palavras diferentes geralmente possuirão *stems* diferentes, cujas pesquisas nos motores de busca retornarão um resultado mais consistente. Orengo *et al.* (2001) afirmam que a utilização de *stemming* normalmente traz benefícios ao processamento de texto para extração de informações.

2.3 *Stopwords*

Stopword é um termo utilizado para designar palavras que são irrelevantes para algum tipo de processamento de texto como, por exemplo, detecção de indícios de plágio. Segundo Arenhardt (2013), são consideradas *stopwords* as classes gramaticais advérbios, artigos, conjunções, preposições e pronomes.

2.4 Computação Paralela

A computação paralela consiste na execução de várias operações simultâneas. Este paradigma de programação está baseado na divisão de trabalho, onde o objetivo é particionar problemas grandes e complexos em problemas menores e mais simples, os quais podem ser executados simultaneamente, trazendo ganho de desempenho.

Foster, (1995) define as características de um sistema de computação paralelo da seguinte forma:

- Um processamento em paralelo consiste de uma ou mais tarefas, que são executadas concorrentemente e podem variar em número ao longo da execução de um programa.
- Tarefas consistem de um trecho de execução sequencial, possuindo seu espaço de memória local.
- Tarefas podem criar novas tarefas, comunicar entre si (envio e recebimento de mensagens) e terminar sua execução.

Existem diversas arquiteturas de programação paralela, dentre as quais, podem ser citadas: arquitetura multicomputador (*cluster*), multiprocessador (multi-CPU), GPU (*Graphics Processing Unit*), e arquiteturas híbridas. Este trabalho tem seu foco voltado na aplicação de arquitetura de *cluster* e Multi-CPU, os quais serão retratados a seguir.

2.4.1 Arquitetura Multicomputador (*cluster*)

Este tipo de arquitetura de processamento paralelo é caracterizada pela utilização de mais de um computador (também chamado de nó) que executam tarefas menores, colaborando entre si para a realização de uma tarefa maior, trazendo ganho de desempenho. Segundo Foster (1995), cada computador é conectado à rede de intercomunicação e executa seu próprio programa. Esse programa possui acesso a sua memória local e pode trocar mensagens com

programas que estão executando em outros nós para trocar informações ou obter acesso à memória remota. A figura 1 apresenta esse modelo de arquitetura:

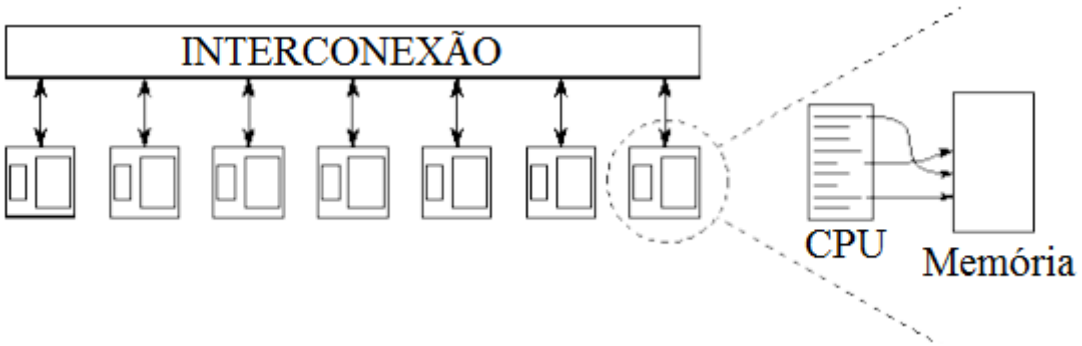

Figura 1 – Exemplo de estrutura multicomputador. Cada nó de execução é composto de sua CPU e memória, e é conectado a uma rede de intercomunicação. Adaptado de Foster, (1995, p. 11).

2.4.2 Arquitetura Multiprocessador (multi-CPU)

Multiprocessador é o nome dado a um tipo de arquitetura paralela, onde um computador possui mais de um processador. Nessa arquitetura, os processadores compartilham a mesma memória local através de um barramento e cada processador pode executar um programa separadamente. A figura 2, representa um exemplo de arquitetura multi-CPU.

De acordo com Foster (1995), sistemas multiprocessador podem ser utilizados como se fossem um sistema multicomputador. Ou seja: pode-se criar diversos processos para um determinado programa, os quais serão executados simultaneamente em diversos processadores do mesmo nó, trocando mensagens entre si. Desta forma, diversas tarefas podem ser executadas por diferentes processos ao mesmo tempo, resultando em ganho de desempenho de execução do programa.

Porém, existe uma abordagem mais simples para a utilização da arquitetura multiprocessador, que é a implementação baseada em *threads*. *Thread*, também conhecido como "processo leve" é um termo utilizado para designar um fluxo de execução independente de um programa.

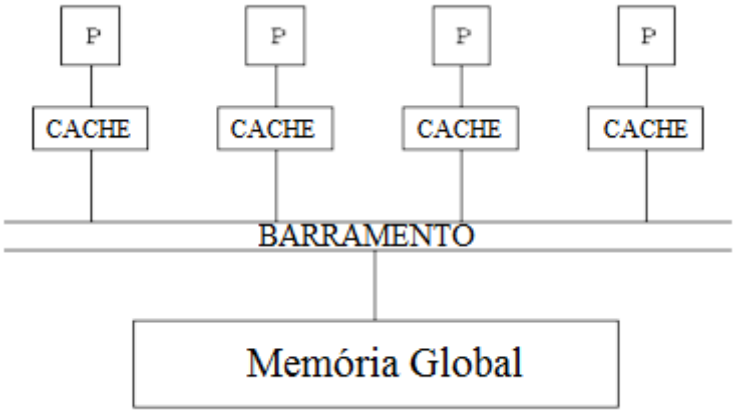

Figura 2 – Exemplo de arquitetura multiprocessador. Cada processador possui acesso à memória principal através de um barramento, e executa seu próprio trecho de código. Adaptado de Foster (1995, p. 12).

Um programa que faz a utilização de várias *threads* é comparável a um programa que faz uso de diversos processos, porém o uso de *threads* traz vantagens como maior simplicidade na troca de informações, além da maior simplicidade de sua criação e destruição, comparado ao uso de processos (TANEBAUM, 2001). Portanto, ao invés de se criar mais de um processo para tirar proveito da existência de vários processadores em um nó de execução, é possível criar diversas *threads* pertencentes a um único processo, facilitando assim a troca de mensagens e o compartilhamento de dados entre os trechos de execução do programa.

2.5 *Profiler*

Profiler é o nome dado a uma ferramenta capaz de traçar o perfil de execução de um código. Através de seu uso, é possível obter informações sobre, por exemplo, a quantidade de vezes que um trecho de código é executada, consumo de memória de um determinado programa e tempo de processamento consumido. Conforme é afirmado por Foster (1995), é muito útil para identificar os pontos passíveis de paralelização em um código fonte e realizar a análise dos resultados obtidos com

aquele processo. Exemplos de *profilers* disponíveis para aplicações desenvolvidas na linguagem de programação Java são o NetBeans Profiler[2] e JProfiler[3].

2.6 Miss Marple e Seu Funcionamento

Miss Marple é um *software* voltado para a detecção de indícios de plágio em documentos digitais, capaz de realizar a análise de plágio dos tipos direto e mosaico. Trata-se de uma aplicação *desktop*, sendo implementado através da linguagem de programação Java. Foi desenvolvido com base no método DIP - Detector de Indícios de Plágio (ARENHADT, 2013), (PERTILE, 2011). O funcionamento desse método consiste simplificadamente dos seguintes passos:

- O usuário carrega na aplicação um documento de texto digital, em formato DOC, DOCX ou PDF. Em seguida, o programa executa a leitura do arquivo e armazena o conteúdo lido como texto plano. Gravuras não são incluídas no processo de análise. Em seguida, é realizado um pré-processamento do texto que será vistoriado.
- Os parágrafos do conteúdo lido são divididos em pequenos trechos chamados *tokens*, os quais são enviados para uma busca na Internet utilizando a API de buscas *Google Search Ajax*.
- Os resultados similares encontrados na pesquisa *on-line* são retornados à aplicação e em seguida, é realizado um cálculo de similaridade e posteriormente é exibido ao usuário um relatório dos resultados de análise.

A figura 3, a seguir, auxilia na compreensão dos passos acima descritos.

[2] Disponível em https://profiler.netbeans.org, acesso em dezembro de 2015.
[3] Disponível em https://www.ej-technologies.com/products/jprofiler/overview.html, acesso em dezembro de 2015.

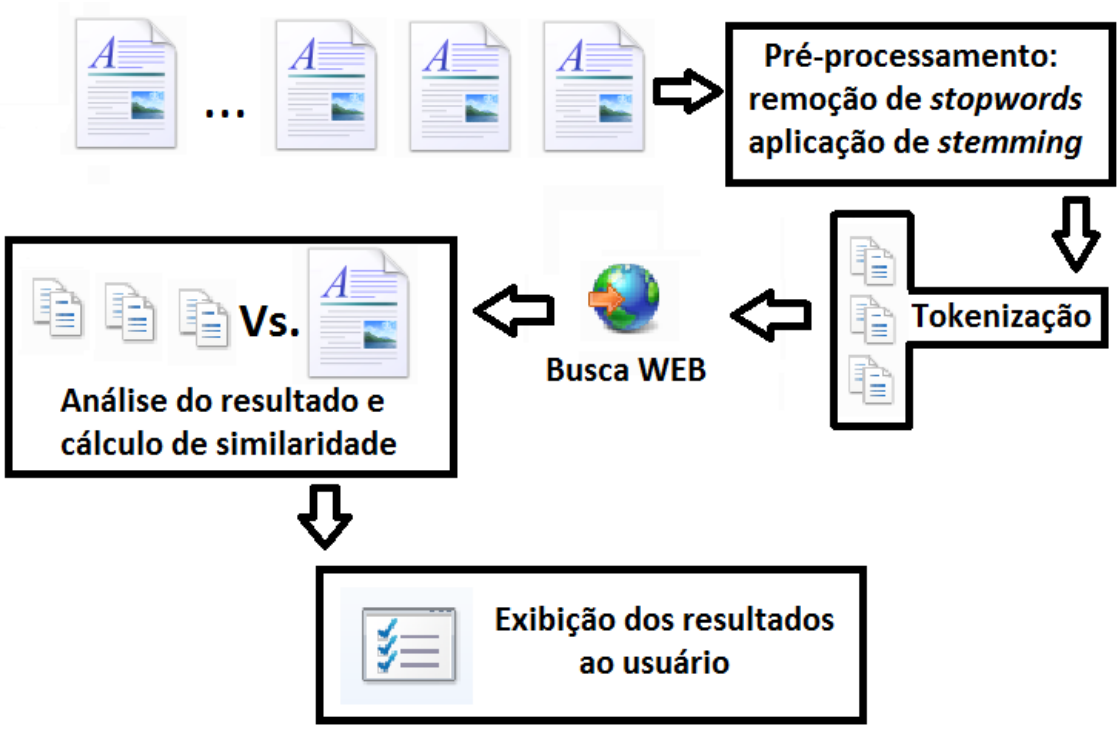

Figura 3 - Estrutura de funcionamento do Miss Marple. Fonte: próprio autor.

No caso específico encontrado no programa Miss Marple, as etapas de pré-processamento e análise de resultados são realizadas através da biblioteca Apache Lucene. Na etapa de pré-processamento, são realizadas as tarefas de remoção de *stopwords* e aplicação de técnicas de *stemming*. Esta última etapa não é executada no método DIP original.

Após realizar o processo de busca, o programa executa o cálculo de similaridade entre os trechos do documento que está sendo verificado e os trechos recuperados na pesquisa, através de um algoritmo chamado Similaridade Cosseno. O método responsável por esta etapa retorna um percentual de similaridade encontrado, o qual é exibido para o usuário.

Arenhardt (2013) e Pertile (2011) definem 60% como sendo o valor máximo de similaridade aceitável para cada parágrafo de um documento. Portanto, valores superiores a 60% de similaridades apontam indícios de ocorrência de plágio. Vale ressaltar que embora o *software* aponte indícios, é papel do usuário efetuar a confirmação da real existência de plágio no documento.

2.7 Trabalhos correlatos

Para o embasamento deste trabalho, foi executado um mapeamento sistemático da bibliografia, detalhado no capítulo da metodologia, a fim de se obter o estado da arte na área de verificação de plágio em documentos digitais e identificar quais trabalhos recentes melhor se assemelham a este. Os resultados desta etapa serão discutidos a seguir.

2.7.1 AntiPlag

AntiPlag é uma ferramenta de análise de plágio em textos digitais, desenvolvido por Jiffriya *et. al.* (2013). Sua arquitetura é bastante similar àquela encontrada no aplicativo Miss Marple.

O primeiro ponto similar entre as duas ferramentas é o foco de aplicação: ambos se destinam à análise de documentos de texto digitais. Segundo, o arquivo lido por ambas aplicações é armazenado internamente como texto plano. Em seguida é realizado um pré-processamento do texto onde, assim como no Miss Marple, gravuras são desconsideradas e as *strings* são padronizadas em letras minúsculas.

As principais diferenças encontradas entre as duas ferramentas se dão após o estágio de pré-processamento. A primeira delas se encontra na etapa de *tokenização* / agrupamento. Enquanto o Miss Marple constrói grupos de até trinta *tokens* em um único parágrafo, o AntiPlag gera diversos tri-gramas a partir de cada frase lida. Em seguida, a ferramenta AntiPlag executa um algoritmo de similaridade diretamente entre os tri-gramas do documento que está sendo verificado e um documento-base de comparação.

Já no caso do Miss Marple, os agrupamentos de *tokens* passam por um processo de *stemming* e após, são enviados para o motor de busca na Internet. Somente então, os resultados de busca são baixados e comparados com conteúdo lido.

2.7.2 *Plagiarism Detection by Identifying the Keywords*

Nesse trabalho, Dutta *et. al.* (2014) apresentam uma forma de detectar documentos *possivelmente* plagiados, a partir de suas palavras-chave. Nele, um arquivo suspeito, em formato PDF é carregado. Em seguida, a primeira página do documento é convertida para uma imagem em formato JPG, onde é aplicado um algoritmo de reconhecimento de padrões.

Na saída desse algoritmo, encontram-se as palavras-chave encontradas no documento carregado, as quais são pesquisadas em um banco de dados à procura de sinônimos. O resultado dessa etapa são as palavras-chave do documento suspeito, juntamente com seus respectivos sinônimos. Por fim, todos esses termos são comparados com as palavras-chave de um documento previamente conhecido como original. Na ocorrência de semelhança entre os termos do documento original com aqueles extraídos do processo anterior, o documento verificado é classificado como "suspeito de ser plágio".

A análise das etapas acima descritas revelam algumas limitações. Primeiramente, o arquivo a ser analisado deve necessariamente ser em formato PDF. Segundo, analisar a existência de plágio somente através das palavras-chave pode levar a conclusões errôneas, dado que todo o restante do não é processado.

No caso do Miss Marple, existe uma certa flexibilidade em questão ao formato de arquivo a ser analisado, sendo suportados também os formatos DOC e DOCX. Também, deve-se notar que o Miss Marple analisa todo o conteúdo do texto, trazendo resultados mais precisos.

2.7.3 Um Modelo Conceitual para Desenvolver uma Avançada Ferramenta de Análise de Plágio Baseado em Análise Semântica

Nesse trabalho, Sarkar *et. al.* (2014) propõem um novo modelo de sistema de detecção de plágio, baseado em análise semântica. O sistema proposto é constituído da seguinte forma:

1. Primeiramente, ocorre a leitura do documento de texto.
2. Realização de uma classificação do tema abordado no documento submetido, no intuito de restringir o espaço de busca das operações seguintes.
3. Uma vez classificada a área temática, é realizada a separação das seções do texto (capa, resumo, introdução, referências, etc.).
4. Filtragem dos trechos candidatos: nesta etapa, o sistema removeria trechos do texto, separados anteriormente, onde não se faz necessária a análise visando busca de plágio como, por exemplo, informações do autor e a seção de referências bibliográficas.
5. Verificação de plágio de forma sintática (baseada em similaridade entre *strings*) e de forma semântica (levando em conta também, o significado das palavras e seus sinônimos).
6. Armazenamento dos resultados da etapa anterior em uma base de dados.
7. Cálculo do percentual geral de similaridade.
8. Classificação do documento entre não plagiado ou plagiado, baseado no cálculo realizado na etapa anterior.

Como é possível de se observar, a arquitetura proposta por Sarkar *et. al.* (2014) possui algumas etapas que são comuns a Miss Marple, sendo elas: carregamento do texto, verificação de plágio de forma sintática e cálculo de similaridade. As etapas 2, 3 e 4 e 6 não são abordadas na atual implementação do Miss Marple.

Ainda, na etapa de verificação de plágio existem diferenças. A arquitetura proposta pelo autor faria uso de um extenso banco de dados, contendo palavras e seus respectivos sinônimos e significados. Enquanto aquela arquitetura leva em consideração a semântica, o Miss Marple não o faz.

Também haveria um banco de dados somente para armazenamento dos resultados de análise, o que não ocorre neste trabalho. Porém, manter um extenso banco de dados pode se tornar inviável para uma aplicação *desktop*. Desta forma, o Miss Marple se destaca, por ser menos oneroso computacionalmente, e ainda assim possuir bons resultados de análise, conforme afirmado anteriormente.

2.8 Diferencial Deste Trabalho

A seção 2.7 trouxe uma comparação entre os trabalhos correlatos e o Miss Marple em sua primeira versão. Também foi retratado anteriormente que este trabalho tem por objetivo implementar melhorias àquele *software*, através uma abordagem de computação paralela e distribuída. A estrutura básica de análise e detecção de indícios de plágio, a qual foi tratada na seção 2.6, bem como as APIs utilizadas na primeira versão do programa, permanecem.

Desta forma, este trabalho herda os diferenciais do Miss Marple em relação aos demais, além de ter somado a si o diferencial de proporcionar ao usuário melhor tempo de processamento de grandes lotes de documentos, através do uso de técnicas de computação paralela e distribuída.

3 METODOLOGIA

Este trabalho se originou através da utilização experimental da aplicação de busca por indícios de plágio Miss Marple, desenvolvida anteriormente no Grupo de Redes e Computação Aplicada. Após a percepção de pontos limitantes e com bom potencial de aprimoramento, o foco de atuação do trabalho foi definido através de uma reunião em conjunto com a equipe responsável pela execução do projeto anterior.

Após definido o tema de atuação, deu-se início a um processo de busca sistemática na bibliografia, também chamado de Mapeamento Sistemático (MS). Esta etapa é descrita detalhadamente na seção 3.1 deste capítulo.

Uma vez realizado essa etapa, partiu-se para a determinação do ferramental necessário para a execução do projeto. Primeiramente, a nova proposta de sistema foi modelada através do uso do padrão UML (Unified Modelling Language - Linguagem de Modelagem Unificada). Em seguida definiu-se a linguagem de programação Java como o padrão para o desenvolvimento do código-fonte, uma vez que o *software* Miss Marple é implementado em Java e esta linguagem permite que o aplicativo seja executado em diversas plataformas computacionais.

Determinada a linguagem, foi definida a IDE (Integrated Development Environment) de trabalho como sendo o NetBeans versão 8.0.2. A escolha se deu devido ao fato desta ser uma ferramenta gratuita e bem difundida.

Em seguida, foram estabelecidas as arquiteturas de paralelismo que seriam aplicadas. Foram escolhidas as arquiteturas multiprocessador multicomputador, devido ao fato de que ambas permitem o uso de *hardware* doméstico, que por sua vez, é mais acessível aos usuários. É mister lembrar que esta escolha não elimina a possibilidade de se utilizar um *hardware* de alto nível.

A fim de paralelizar do programa em nível multi-CPU, optou-se por realizar uma implementação baseada em *threads*. Já no intuito de paralelizar a aplicação em nível de multicomputador, optou-se pela utilização da API Java *Remote Method Invocation* (RMI) para a troca de mensagens entre os processos.

Para a execução de testes do sistema desenvolvido foram utilizados 2 computadores conectados a uma rede local por cabo de rede, com velocidade de

conexão de 100Mbps. As configurações de *hardware* dos equipamentos são as seguintes:

- Um Intel Core2 Quad Q6600 2.4GHz com 4GB de memória RAM.
- Um Intel Core i7 860 (quad-core) com Hyper Threading (HT) ativado (2 threads por núcleo físico) 2.8GHz com 8GB de memória RAM.

As duas máquinas foram configuradas com o sistema operacional Microsoft Windows 7 Professional de 64 bits.

Para fins de testes dos resultados e validação do sistema, fez-se uso da ferramenta *profiler* nativa do NetBeans. Primeiramente foi executado o *profiler* utilizando-se a aplicação Miss Marple na versão não paralelizada para um conjunto de 24 documentos com tamanho variando de 300KB até 3MB. Os dados obtidos foram confrontados com o processamento do mesmo lote de documentos pela versão paralelizada do Miss Marple. Os resultados são apresentados no capítulo 5 deste trabalho.

3.1 Mapeamento Sistemático

A definição de mapeamento sistemático, por Kitchenham (2014) é "uma metodologia com etapas bem estruturadas, capaz de proporcionar o correto embasamento, necessário para a realização de uma boa revisão bibliográfica".

O objetivo desta revisão sistemática foi determinar o estado da arte no que diz respeito ao processamento e verificação de indícios de plágio. As questões utilizadas para nortear a pesquisa foram:

- Como se dá o processo de busca por indícios de plágio em documentos digitais e quais as semelhanças entre as demais aplicações e o *software* Miss Marple?
- Em que etapas o paralelismo pode ser utilizado para aprimorar este processo?
- Quais os trabalhos correlatos que utilizam alguma forma de programação paralela em um sistema de análise de plágio.

Com base nos objetivos traçados para a pesquisa, deu-se início ao processo de definição da *string* de busca. Após diversas tentativas, chegou-se à seguinte *string*, que recuperou os resultados mais próximos aos desejados: *parallelize* OR *parallel* OR *plagiarism* OR "*check plagiarism*" OR "*plagiarism detection*". As palavras-chave foram fixadas como sendo em inglês, uma vez que as primeiras tentativas de busca demonstraram que os resultados mais relevantes foram encontrados através de palavras-chave em inglês.

Em seguida, foram delimitadas as seguintes bases para se efetuar a pesquisa: IEEE Xplore, Jurn e Scielo. A escolha destas se deu ao fato de possuírem bom respaldo na área de ciência da computação e tecnologia. Para a execução, foi utilizado o motor de busca Google Scholar. Os resultados obtidos foram salvos e organizados através do aplicativo Mendeley[4].

Primeiramente, a busca retornou um total de 198 artigos na base IEEE Xplore, nenhum resultado na base Jurn, e 11 resultados na base Scielo. A partir de então, partiu-se para o processo de filtragem refinada dos resultados, tomando-se os seguintes critérios de inclusão e exclusão:

- Inclusão: artigos que tratem sobre o processo de detecção de plágio em documentos de texto digital e/ou uso de paralelismo neste processo.
- Exclusão: artigos que não estejam relacionados com os critérios acima ou cujos objetivos não se demonstraram claros, ou artigos que estejam em idioma diferente do português, inglês e espanhol.

Vale ressaltar que embora os idiomas português e espanhol não tenham sido incluídos na *string* de busca, esses se mantém válidos, a fim de incluir resultados contendo o corpo do texto escrito nestes idiomas.

Após a etapa de refinamento dos resultados, foram retidos um total de 38 artigos. Por fim, foram identificados os 3 artigos mais similares a este trabalho, os quais foram detalhados na seção de trabalhos correlatos, do capítulo anterior.

[4] Disponível em <https://www.mendeley.com>, acesso em setembro de 2015.

4 PROPOSTA: PARALLEL MISS MARPLE

Conforme exibido no início da apresentação deste trabalho, foi identificado que a execução sequencial do código de um programa de verificação de indícios de plágio pode se mostrar um fator limitante de sua aplicação em documentos extensos ou grandes lotes de documentos. Um dos motivos deve-se ao fato de que o potencial do processador não é explorado ao máximo, acarretando demora excessiva na análise de indícios de plágio. A figura 4, a seguir, exemplifica este fato.

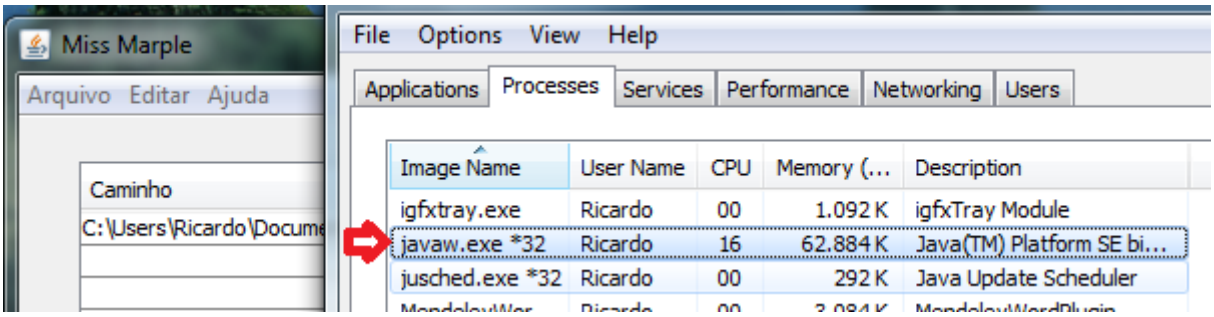

Figura 4 – Baixo aproveitamento de CPU do *Miss Marple* original. Fonte: próprio autor.

Miss Marple original, executando em um computador com processador Intel Core i3 2310M e 6GB de memória, com sistema operacional Microsoft Windows 7 Professional 64 bits.

No intuito de mitigar esta questão, propõe-se o desenvolvimento de uma nova versão do *software* de busca por indícios de plágio Miss Marple, batizada de *Parallel Miss Marple*. Nesta nova versão, o aplicativo recebe a capacidade de extrair melhor proveito das arquiteturas multiprocessador dos computadores atuais, tornando a verificação de plágio mais rápida, sem comprometer a precisão dos resultados.

4.1 Modelagem da Ferramenta

"A modelagem possibilita que o desenvolvedor projete as funcionalidades e ações do *software* e do usuário, pois possibilita a visualização e a comunicação entre o desenvolvedor e o usuário a partir de diagramas." (ARENHADT, 2013) *apud* (IBM, 2013).

Para o auxílio na definição das ações esperadas da ferramenta e do usuário, foram construídos os diagramas de casos de uso do aplicativo e do utilizador, através da linguagem UML. Os diagramas são apresentados pelas figuras 5 e 6.

Figura 5. Casos de uso usuário/ferramenta. Fonte: próprio autor.

A figura 5 representa as ações do usuário perante o aplicativo. Dentro da ferramenta, utilizador seleciona os arquivos que devem ser analisados, configura parâmetros de busca na Internet, configura opções avançadas de processamento e solicita o início da análise. Em seguida, verifica os relatórios gerados pelo aplicativo, assim como os arquivos salvos no repositório.

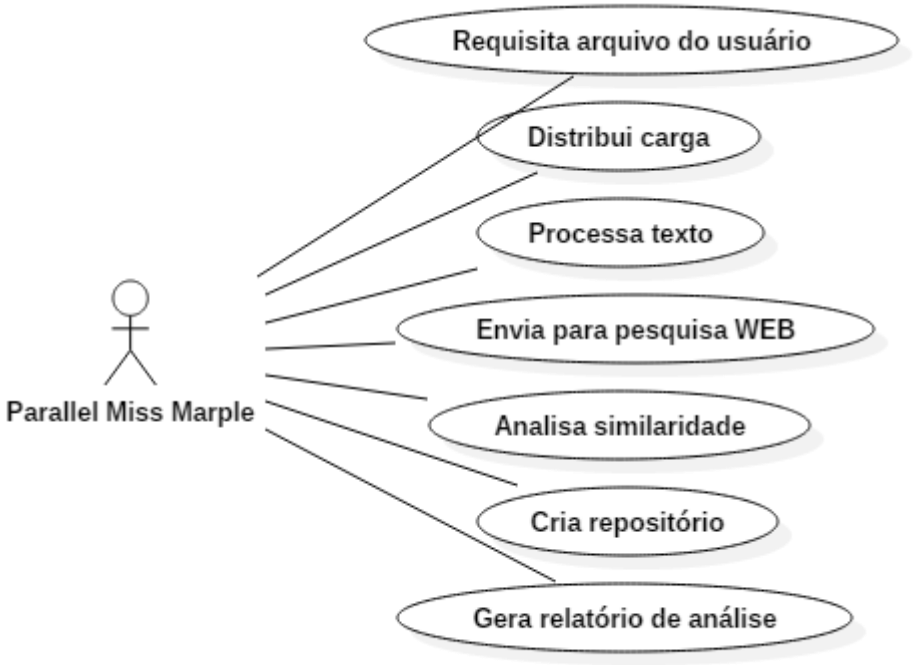

Figura 6. Casos de uso da ferramenta. Fonte: próprio autor.

No diagrama da figura 6 observam-se as ações do aplicativo *Parallel Miss Marple* ao analisar um conjunto de arquivos. A ferramenta requisita arquivos ao usuário. A seguir, o aplicativo distribui as tarefas entre *threads* de análise e/ou outros computadores na rede, de acordo com a configuração do usuário. Cada *thread* executa o pré-processamento do texto e uma posterior busca na WEB. Os resultados obtidos na busca passam por uma análise de similaridade. Arquivos com um percentual de similaridade superior a 60% são salvos em um repositório local. Por fim, é gerado um relatório de cada análise efetuada.

4.2 Arquitetura do *Parallel Miss Marple*

A arquitetura da ferramenta proposta é composta de quatro principais componentes: módulo de entrada de arquivos, *threads* de análise (ou analisadores), módulo de controle e distribuição de carga, módulo de interface remota e módulo de interface do usuário. Suas funções e funcionamento são detalhadas a seguir.

4.2.1 Módulo de Entrada de Arquivos

Possui como função o carregamento de documentos de texto digital em uma fila de análise e sua posterior conversão para texto plano. São suportados os formatos DOC, DOCX e PDF.

4.2.2 *Threads* de Análise ou Analisadores

Responsável por realizar a análise de plágio nos documentos como um todo. Cada *thread* de análise executa os mesmos passos encontrados no Miss Marple original (remoção de *stopwords*, *tokenização*, busca WEB, análise de resultados e cálculo de similaridade). Ou seja: cada analisador pode ser visto como se fosse uma instância independente do Miss Marple original.

A figura 7, a seguir, representa o trabalho de análise de um documento realizado por um analisador. Como pode ser visto, os passos são basicamente os mesmos que são encontrados na figura 3 do capítulo 2.

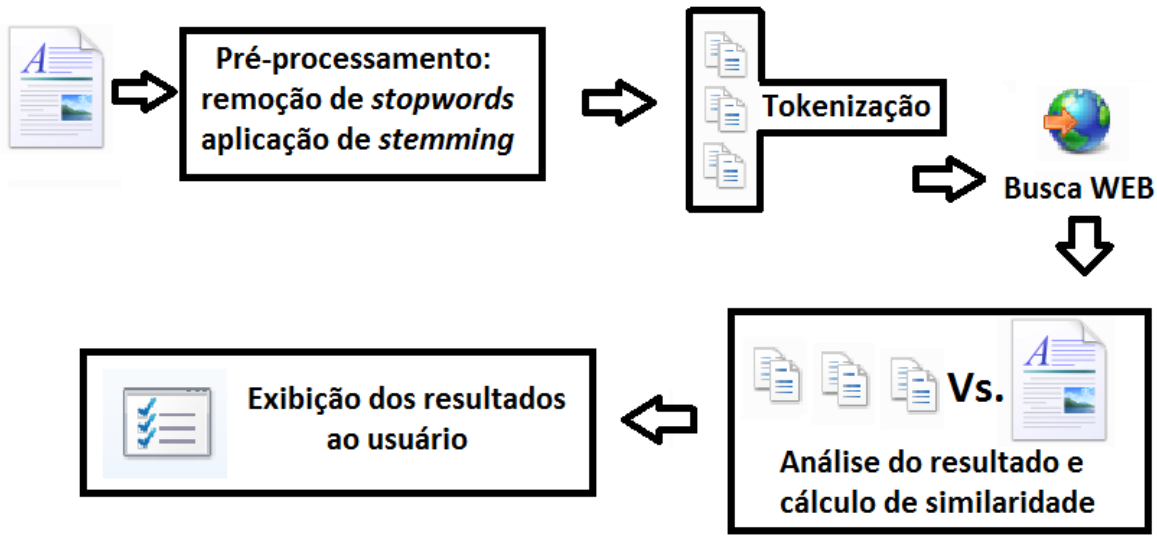

Figura 7 - Trabalho de processamento de uma *thread* no *Parallel Miss Marple.* Fonte: próprio autor.

Sendo a figura 7 a representação de uma *thread* de análise de documentos, o sistema pode ser visto de forma mais ampla na figura 8, no caso de execução em apenas um computador, e pela figura 9, onde mais de um computador executa o *Parallel Miss Marple* para a divisão de carga.

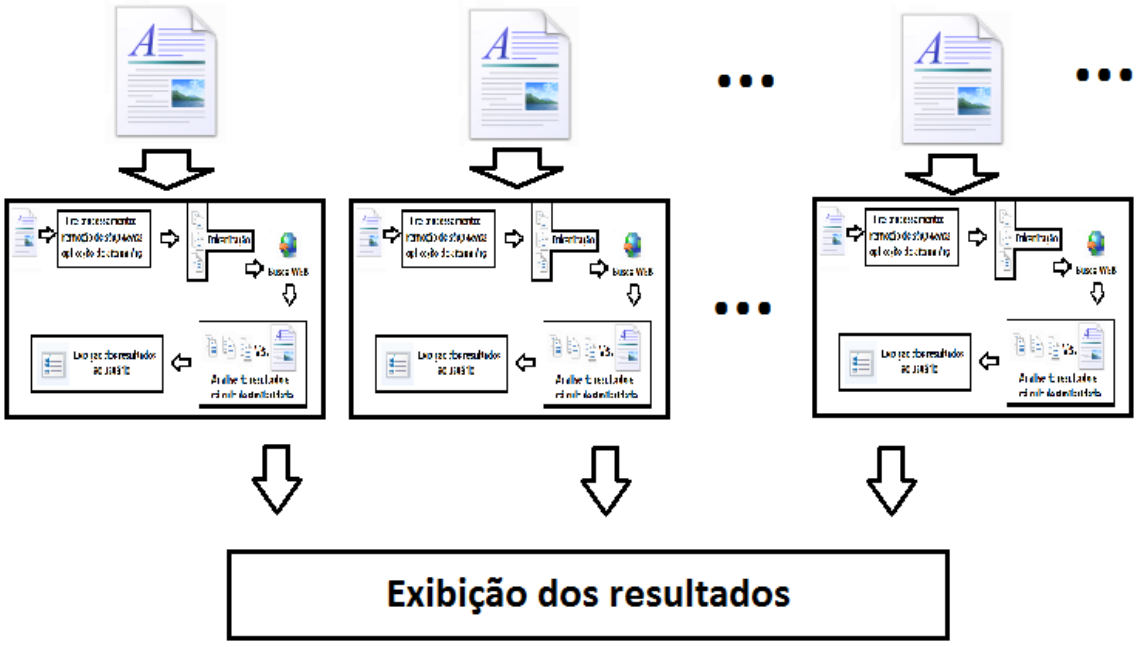

Figura 8 - Visão geral do sistema *Parallel Miss Marple*, em um computador
Fonte: próprio autor.

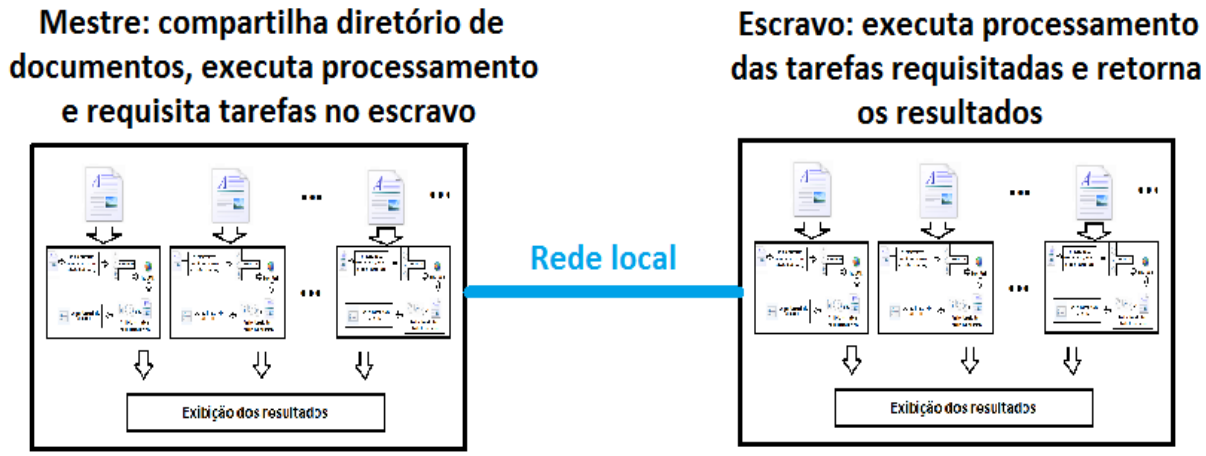

Figura 9 - *Parallel Miss Marple*, distribuído entre duas estações de trabalho.
Fonte: próprio autor.

4.2.3 Módulo de Controle e Distribuição de Carga

Responsável pela criação de *threads* de análise e distribuição de documentos dentre elas. Também é responsável pela distribuição de carga entre outros computadores executando *Parallel Miss Marple* na rede. Uma vez que o programa é iniciado, o módulo de controle possui dois modos de operação possíveis, configuráveis pelo usuário: modo mestre ou modo escravo. O aplicativo não pode ser configurado como mestre e escravo ao mesmo tempo .

4.2.3.1 Modo Mestre

Este modo de operação é responsável pela criação dos analisadores e distribuição das tarefas de análise. Para isto, são instanciados objetos do tipo analisador: `Analisador analisador = new Analisador(parâmetros)`. O número de analisadores criados variam de acordo com a configuração especificada pelo usuário, que pode ser dividida em três casos:

- Configuração 1: número ilimitado de *threads*. Nesta configuração, o programa cria um analisador para cada documento submetido.
- Configuração 2: número de analisadores igual ao número de processadores. Nesta configuração, o *software* detecta, em tempo de execução, o número de núcleos de processamento existentes, e cria um analisador para cada núcleo detectado.
- Configuração 3: O usuário especifica o número de *threads* que devem ser criadas.

Uma vez criadas as *threads*, o controlador faz uso de um escalonador *Round-robin* para a distribuição de tarefas entre os analisadores. Em seguida, o controlador ordena a cada analisador o início do processamento através do método `analisador.execute()`.

Vale ressaltar que o escalonador do *Parallel Miss Marple* é responsável pela distribuição das tarefas de análise entre as *threads* analisadoras. O escalonamento

das *threads* criadas por entre os processadores do computador é executado pelo sistema operacional.

O módulo de controle também é responsável pela distribuição de tarefas entre outros computadores na mesma rede, que estejam executando o programa. Quando a distribuição entre computadores na rede está ativada, o programa cria uma *thread* para cara arquivo encontrado na lista de análise. Para isso, o controlador faz uma solicitação de análise remota através da API Java RMI. Primeiro, é realizada é criado uma instância do objeto de interface remota:

```
IMissMarple imm = (IMissMarple)Naming.lookup("rmi://"+
addresses.get(i)+ "/IMissMarple").
```

Em seguida, o mestre coloca o arquivo que será analisado remotamente disponível para ser transferido para o computador escravo, através de uma conexão *socket*, tarefa executada pela classe *FileSender*:

```
FileSender fs = new FileSender(arquivo);
fs.start();
```

Finalmente, é solicitada a análise remota através do comando `imm.analise(arquivo, parametros_de_busca)`. Este comando é processado no computador escravo e através dele, são informados todos os parâmetros necessários para o processo de busca.

4.2.3.2 Modo Escravo

Neste modo de operação, o controlador aguarda uma solicitação de análise remota vinda de um computador executando o programa no modo mestre. Uma vez recebida a solicitação, o controlador recebe do mestre uma cópia do arquivo a ser analisado através de uma conexão de rede via *socket*. Em seguida, o arquivo é processado localmente. Ao término do processamento, uma mensagem é retornada ao mestre, indicando a conclusão da tarefa.

4.2.4 Interface do Usuário

Promove a interação entre a ferramenta e o usuário, permitindo a submissão de arquivos, visualização dos resultados e a configuração de parâmetros de busca, tais como: número máximo de pesquisas na WEB para cada documento submetido e tipos de arquivos que podem ser baixados das fontes na Internet. A figura 10 mostra a interface principal da aplicação, enumerando seus principais componentes.

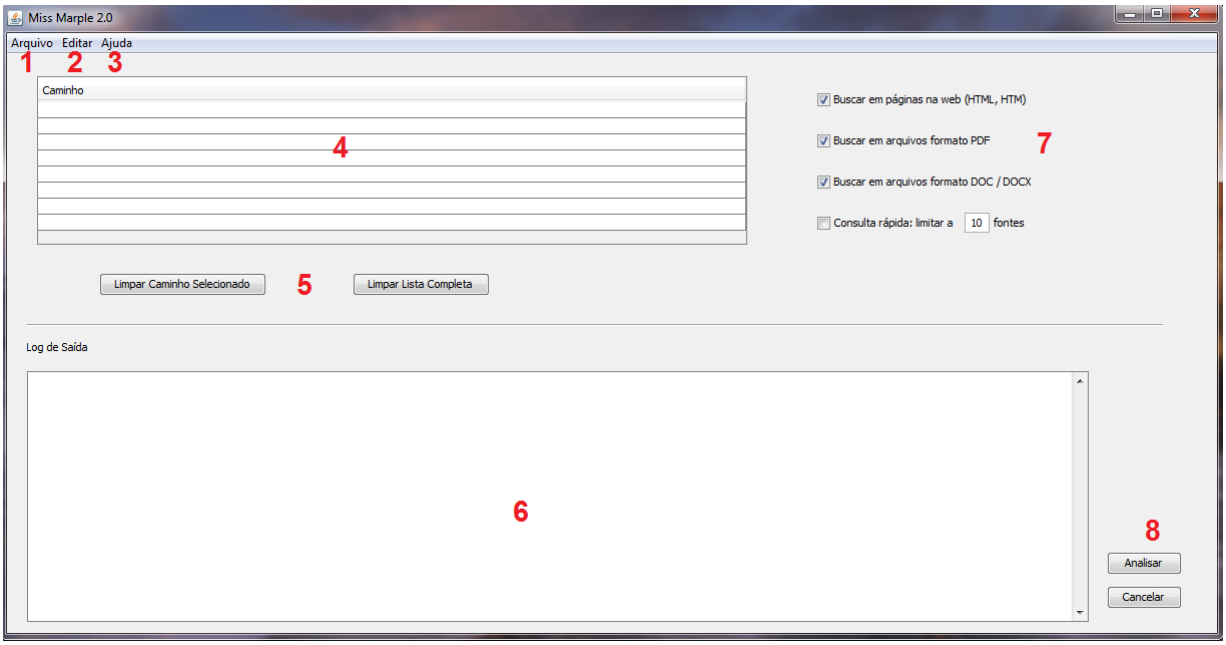

Figura 10. Tela principal da aplicação. Fonte: próprio autor.

Os componentes enumerados são:

1. Menu "arquivo": por meio deste, o usuário pode carregar arquivos para análise no programa e acessar o repositório criado durante a execução.
2. Menu "editar": possibilita ao usuário a alteração do diretório do repositório que será criado e acesso às configurações avançadas de processamento.
3. Menu "ajuda": possibilita acesso ao manual do programa.
4. Lista de documentos: exibe todos os documentos que foram submetidos para análise.

5. Botões "limpar o caminho selecionado" e "limpar a lista completa": removem arquivos da lista de entrada do programa.
6. Caixa de saída: exibe em tempo real, informações relativas ao processo de análise dos documentos.
7. Opções de busca: possibilita ao usuário selecionar o tipo de fonte que será consultada na internet (páginas da WEB, arquivos DOC, DOCX e PDF) e limitar a quantidade de consultas realizadas.
8. Botões "analisar" e "cancelar": iniciam ou cancelam o processo de análise. Quando o *Parallel Miss Marple* se encontra no modo escravo, estes botões tem suas ações desativadas.

Já a figura 11, exibe o menu de configurações avançadas de processamento. Nele, são encontradas opções relativas ao processamento do *Parallel Miss Marple*. Seus itens serão detalhados a seguir.

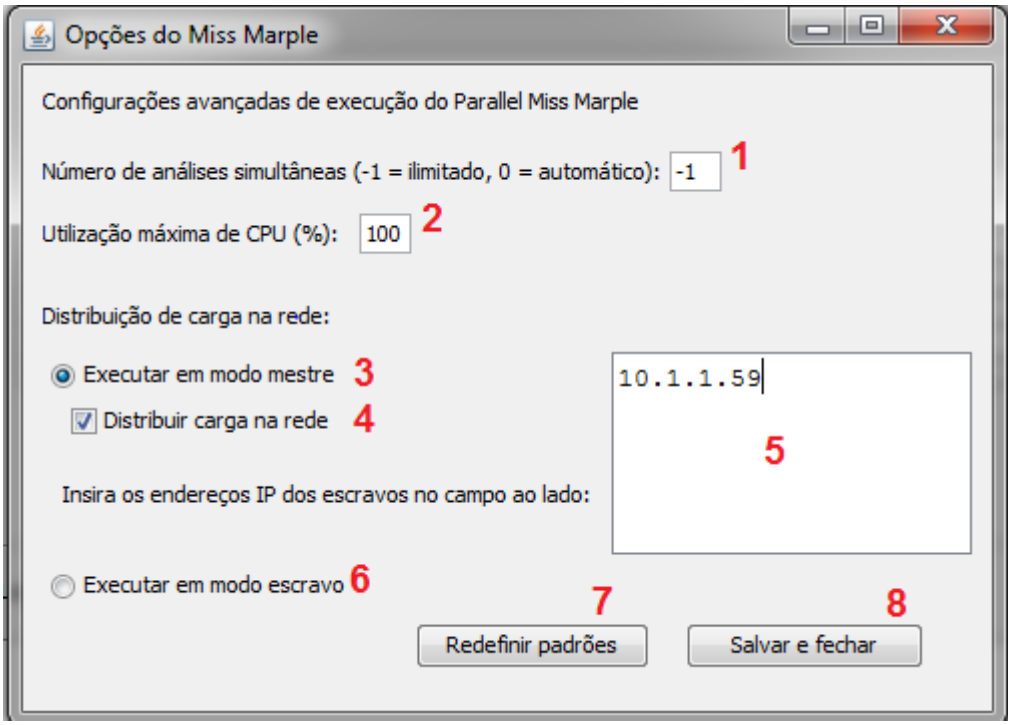

Figura 11. Opções de processamento do *Parallel Miss Marple.* Fonte: próprio autor.

Os campos enumerados são:

1. Número de análises simultâneas: determina o número de *threads* de análise que o programa irá criar. Os valores de entrada podem ser -1, 0 ou algum número superior a zero. No primeiro caso, o programa cria uma *thread* para cada documento. No segundo, o programa cria tantos analisadores quanto forem os processadores do computador. Já no terceiro, o aplicativo cria o número de *threads* informado pelo usuário.
2. Utilização máxima de CPU: determina o máximo de CPU que o programa deve utilizar. Caso este limite seja ultrapassado, o programa evita a execução de novas *threads*.
3. Opção "executar em modo mestre": informa ao aplicativo que ele deve executar em modo mestre, conforme descrito no item 4.2.3.1.
4. Distribuir carga na rede: quando habilitado, o *Parallel Miss Marple* distribui tarefas entre outros computadores rodando o programa.
5. Endereços IP dos escravos: neste campo, o usuário informa os endereços IP dos computadores que devem ser utilizados como escravos.
6. Opção "executar em modo escravo": configura o aplicativo para receber requisições remotas vindas de um mestre, conforme detalhado no item 4.2.3.2. Quando configurado para funcionar neste modo, o programa não aceita submissões locais.
7. Botão "redefinir padrões": redefine todas as configurações de processamento para seus valores padrão. O campo "número de análises" assume o valor 0, o uso máximo de CPU é configurado para 90%, o modo de mestre de operação é ativado e a distribuição de carga na rede é desativada.
8. Botão "salvar e fechar": salva as configurações do usuário em um arquivo de configuração chamado "mmconfig.cfg", para que possam ser recuperadas em execuções posteriores do programa. Em seguida, fecha o menu de configuração.

Uma vez executada a análise dos documentos, o programa exibe uma mensagem de notificação para o usuário, conforme apresentado pela figura 12. Em seguida, o utilizador pode abrir o diretório do repositório, onde são encontrados o relatório de análise e os arquivos recuperados da Internet.

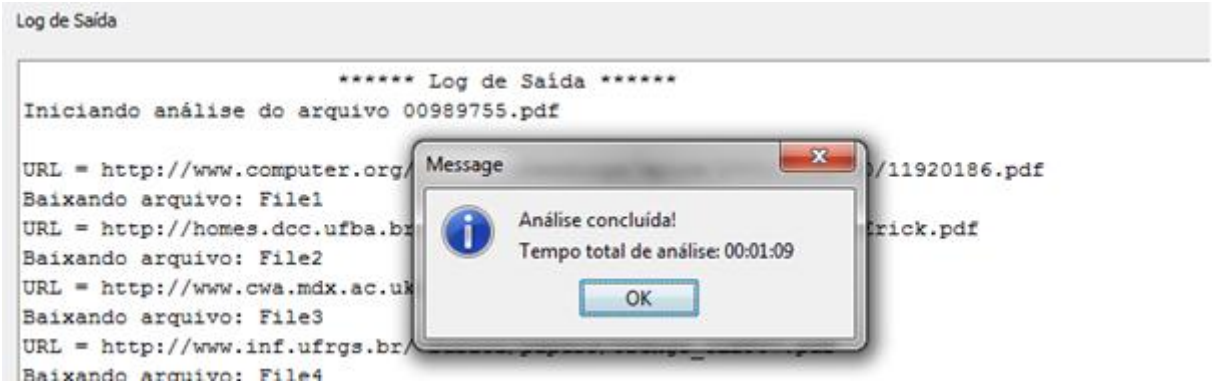

Figura 12. Notificação de conclusão de análise. Fonte: próprio autor.

4.3 Desafios Encontrados

Ao longo do desenvolvimento do trabalho e do período de testes, foram encontradas alguns desafios.

O primeiro deles e já existente na versão inicial do Miss Marple, gira em torno da utilização da API *Google Search Ajax*, pois esta possui um limite de consultas diárias. Uma forma encontrada para mitigar esta questão é a ativação do limite de consultas através da interface do usuário.

Outro fator limitante é a velocidade e estabilidade da conexão com a Internet, que ainda causam impacto no tempo total necessário para realizar a análise dos documentos. Novamente, a limitação do número de consultas pode ser um aliado para minimizar este ponto.

Por fim, assim como o Miss Marple original, não é possível executar o *Parallel Miss Marple* em sistemas operacionais sem interface gráfica. Uma alternativa para solucionar esta questão seria a criação de uma nova versão do aplicativo, de forma a remover a dependência da interface gráfica, e proporcionando a execução através de linha de comando.

5 RESULTADOS

Este capítulo aborda os resultados obtidos durante o processo de validação da ferramenta *Parallel Miss Marple*, comparando-a com a versão anterior do *software*.

5.1 Validação do Aplicativo

Conforme mencionado no capítulo de Metodologia deste trabalho, para validação da ferramenta, fez-se uso do *NetBeans Profiler* para aferir o tempo de processamento das versões serial e paralela do programa. Os testes foram executados utilizando-se como base um conjunto de 24 documentos digitais.

Foram executados três aferições de tempo de processamento da versão original do *Miss Marple*, do *Parallel Miss Marple* executando em apenas um computador (Core i7, descrito na seção Metodologia) e do *Parallel Miss Marple* com distribuição de carga entre outro computador (Core i7 + Core2 Quad). O quadro 1 exibe os resultados obtidos pelas aferições de tempo de análise, em segundos. Todos os tempos registrados consideram os tempos de processamento (CPU) e os tempos de comunicação (comunicação via rede, busca na internet).

Teste	*Threads*	Tempo 1	Tempo 2	Tempo 3	Média
Miss Marple original	1	2248,612s	2133,674s	2196,363s	2192,884s
Parallel Miss Marple	1	2158,585s	2075,857s	2239,077s	2157,839s
Parallel Miss Marple	2	1023,956s	1193,045s	1087,457s	1101,486s
Parallel Miss Marple	4	693,292s	665,558s	703,217s	687,355s
Parallel Miss Marple	8	425,719s	401,502s	443,274s	423,498s
Parallel M.M. Distribuído	24	143,446s	155,260s	157,865s	152,190s

Quadro 1. Tempos de processamento *Miss Marple* e *Parallel Miss Marple*.

Como é observado, a execução do *Parallel Miss Marple* com apenas uma *thread* de execução possui desempenho similar ao da versão original do programa. Já com duas ou mais *threads*, observa-se uma redução no tempo necessário para processar o lote de documentos submetidos.

Ainda com posse destes valores, pode ser calculado o *speedup* da versão paralela. *Speedup*, segundo Foster (1995), é uma medida de ganho de desempenho de um algoritmo, dada pela razão entre a execução sequencial e a execução paralela de um código. O *speedup* ideal é de ordem linear, ou seja, cresce na mesma proporção conforme aumentam o número de *threads*. Ao dobrar o número de *threads*, na teoria, dobra-se o desempenho.

A seguir são apresentados um quadro (quadro 1) contendo o *speedup* de acordo com o número de *threads* e um gráfico (gráfico 1), comparativo entre o *speedup* ideal e o resultados obtidos com o *Parallel Miss Marple* executando no computador principal (Core i7).

Threads	1	2	4	8	24 (Distribuído)
Speedup	1	1,9908	3,1903	5,0702	14,408

Quadro 2. *Speedup versus* número de *threads*.

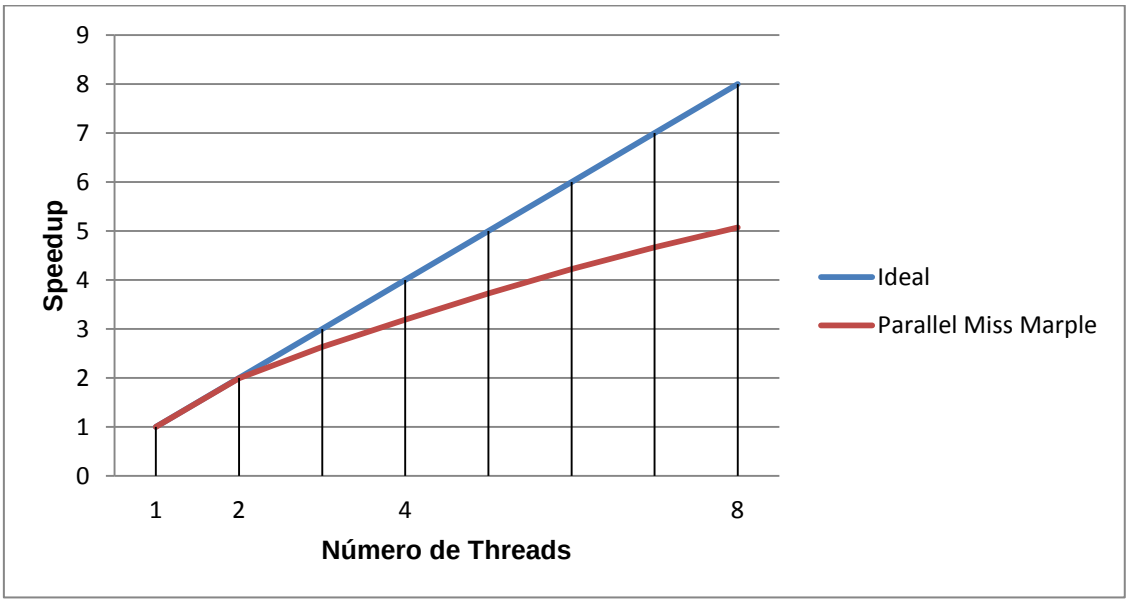

Gráfico 1. *Speedup* ideal *versus* obtidos no trabalho. Fonte: próprio autor.

Analisando-se os dados do quadro 2 e o gráfico 1, é possível perceber um ganho considerável de desempenho através do uso da nova versão da ferramenta. É possível observar que com duas *threads* de análise, o *Parallel Miss Marple* se demonstra bastante próximo do gráfico ideal de melhoria de desempenho. Ou seja, para duas *threads*, o código paralelo é 1,9908 vezes mais rápido do que o sequencial. Para 4 ou mais *threads*, os ganhos ainda são notórios, porém, se distanciam da curva ideal.

Outro ponto a ser analisado é o consumo de memória. Foram amostradas as utilizações deste recurso na versão original do programa e na versão paralela, para fins de comparação. Os resultados são apresentados pelos gráficos 2 e 3 a seguir.

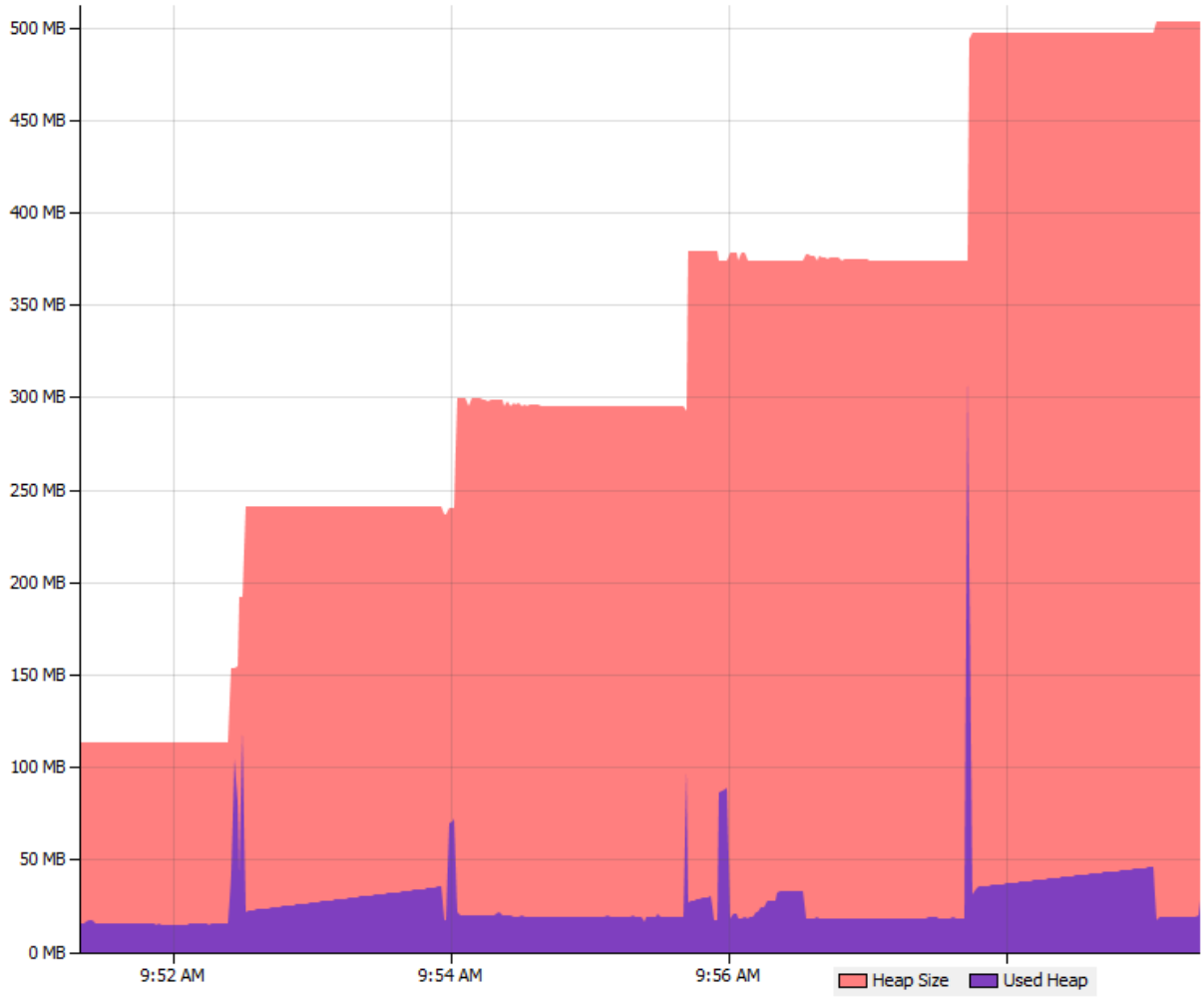

Gráfico 2. Consumo de memória do Miss Marple original. Fonte: próprio autor.

Em vermelho: Memória total alocada. Em roxo: memória utilizada pela aplicação.

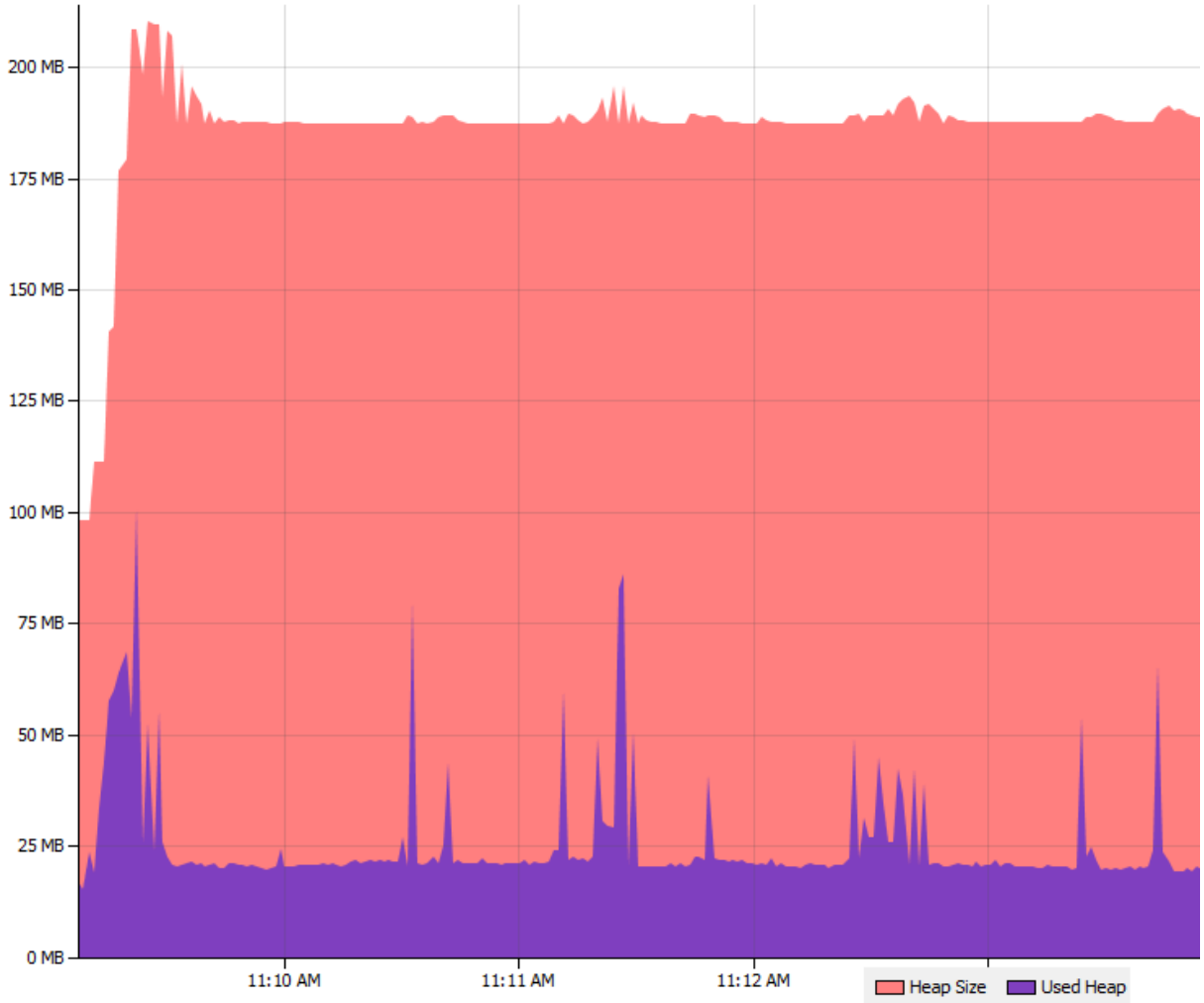

Gráfico 3. Consumo de memória do *Parallel Miss Marple*. Fonte: próprio autor.

Em vermelho: memória total alocada. Em roxo: memória utilizada pela aplicação.

A análise dos gráficos 2 e 3 supra exibidos revela que não houve aumento significativo de utilização de memória na versão paralelizada do aplicativo. Salvos poucos picos de utilização, ambas as versões possuem baixo consumo de memória (em geral, menos de 100MB). Porém, o *Parallel Miss Marple* agrega a este ponto positivo um melhor desempenho, obtido através da utilização de técnicas de paralelismo e distribuição de tarefas. Assim, o tempo necessário para analisar vários arquivos é reduzido.

Contextualizando os resultados, tem-se o seguinte exemplo: um revisor de periódicos deseja realizar a análise em um conjunto de dez artigos recebidos.

Supondo que os arquivos são de mesmo tamanho, ao se fazer uso do *Miss Marple* original, o tempo necessário para análise é de aproximadamente dez vezes o tempo necessário para analisar cada documento, já que se trata de um processo em lote sequencial. Em se tratando de arquivos muito extensos, o tempo total necessário para verificar este pequeno conjunto de arquivos se demonstra demasiadamente custoso, podendo tomar uma hora ou mais.

Porém, ao fazer o uso do *Parallel Miss Marple* em uma configuração de duas *threads* (cenário comum para os computadores atuais), o tempo total necessário pode ser reduzido aproximadamente pela metade. Já com uma configuração de 4 *threads*, o processo pode ser acelerado em até 3,1903 vezes.

Em outro cenário, onde um professor deseja verificar trabalhos de várias turmas, onde o número de arquivos é muito elevado, é possível obter vantagem da distribuição de carga entre outros computadores executando o programa.

Desta forma, o *Parallel Miss Marple* se demonstra com bom potencial de utilização por professores e revisores de trabalhos de periódicos.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo o aprimoramento do *software* de busca por indícios de plágio *Miss Marple* (ARENHARDT, 2013), de forma a mitigar a questão do alto tempo necessário para realizar a análise de grandes lotes de documentos digitais.

Este aplicativo foi desenvolvido no Grupo de Redes e Computação Aplicada da Universidade Federal de Santa Maria, sendo baseado no método DIP - Detector de Indícios de Plágio (PERTILE, 2011) e tendo como objetivo o auxílio no controle de autenticidade de trabalhos acadêmicos. Porém, foi identificado que a análise de grandes lotes de documentos se demonstrava muito oneroso em termos de tempo. Desta forma buscou-se o desenvolvimento de uma nova versão desta ferramenta, no intuito de minorar esta limitação.

Os objetivos propostos neste trabalho foram alcançados, uma vez que os resultados se demonstram satisfatórios, conforme apontam os resultados exibidos na seção 5.

As principais contribuições deste trabalho são: a) redução do tempo total de análise vários documentos; b) suporte a distribuição de tarefas entre outros computadores executando o *software*; c) possibilidade de configuração de parâmetros de processamento através de uma interface gráfica, incluindo parâmetros relativos a paralelismo e a possibilidade de limitar o consumo de CPU pela aplicação; d) salvamento das configurações, para posterior utilização em novas execuções do programa.

Por fim, são sugeridos como trabalhos futuros o desenvolvimento de uma ferramenta com a capacidade de execução em ambientes sem interface gráfica, como aqueles encontrados em servidores, e a criação de um repositório de arquivos localizado em uma nuvem computacional, de forma a reduzir o número de consultas através da API de busca.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ARENHARDT, Catiane Priscila Barbosa - MISS MARPLE - DESENVOLVIMENTO DE FERRAMENTA PARA AUXILIAR NA VERIFICAÇÃO E DETECÇÃO DE INDÍCIOS DE PLÁGIO COM BASE NO MÉTODO DIP - DETECTOR DE INDÍCIOS DE PLÁGIOS. UFSM, 2013

ARENHARDT, C. P. B; MEDINA, R. D.; PERTILE, S. L; GOMES, R. B; TRINDADE, V. L. - Miss Marple: Proposta de Desenvolvimento de Ferramenta de Detecção de Inidícios de Plágio com base no Método DIP - Detector de Indícios de Plágio.

PERTILE, S.L. Desenvolvimento e Aplicação de um Método para Detecção de Indícios de Plágio. UFSM, 2011.

Mendeley. Disponível em www.mendeley.com, acesso em setembro de 2015.

JProfiler. Disponível em www.ej-technologies.com/products/jprofiler/overview.html, acesso em dezembro de 2015.

NetBeans Profiler. Disponível em profiler.netbeans.org, acesso em novembro de 2015.

MORAES, R. O plágio na pesquisa acadêmica: a proliferação da desonestidade intelectual. disponível em www.faculdadesocial.edu.br/dialogospossiveis/artigos/4. Acesso em junho/2015.

KIRKPATRICK, K. Evitando o Plágio, 2001, traduzido por ARQUINO, J. p.2. disponível em http://www.lepem.ufc.br/jaa/plagio.pdf.

KITCHENHAM, B. Procedures for performing systematic reviews. Keele, UK, Keele University, 2004. Disponível em: http://people.ucalgary.ca/~medlibr/kitchenham_2004.pdf, acesso julho 2015.

TANEBAUM , Andrew S. Modern Operating Systems, 2nd Edition, 2001.

FOSTER, I. Designing and Building Parallel Programs, 1995.

ORENGO, V, M., HUYCK, C. A Stemming Algorithm for the Portuguese Language. School of Computing Science Middlesex University, 2001.

JIFFRIYA, M.A.C.; Jahan, M.A.C.A.; Ragel, R.G.; Deegalla, S., "AntiPlag: Plagiarism detection on electronic submissions of text based assignments," in *Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on* , vol., no., pp.376-380, 17-20 Dec. 2013

DUTTA, S.; Bhattacharjee, D., "Plagiarism Detection by Identifying the Keywords," in *Computational Intelligence and Communication Networks (CICN), 2014 International Conference on* , vol., no., pp.703-707, 14-16 Nov. 2014

SARKAR, A.; Marjit, U.; Biswas, U., "A conceptual model to develop an advanced plagiarism checking tool based on semantic matching," in *Business and Information Management (ICBIM), 2014 2nd International Conference on* , vol., no., pp.104-108, 9-11 Jan. 2014