UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

CAIO ARCE NISHIBE

CENTRAL DE CONFRONTOS PARA UM SISTEMA AUTOMÁTICO DE IDENTIFICAÇÃO BIOMÉTRICA: UMA ABORDAGEM DE IMPLEMENTAÇÃO ESCALÁVEL

DISSERTAÇÃO

CURITIBA

2017

CAIO ARCE NISHIBE

CENTRAL DE CONFRONTOS PARA UM SISTEMA AUTOMÁTICO DE IDENTIFICAÇÃO BIOMÉTRICA: UMA ABORDAGEM DE IMPLEMENTAÇÃO ESCALÁVEL

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Ciências" – Área de Concentração: Telecomunicações e Redes.

Orientadora: Prof[a]. Dra. Keiko Verônica Ono Fonseca

CURITIBA

2017

Dados Internacionais de Catalogação na Publicação

Nishibe, Caio Arce
N724c Central de confrontos para um sistema automático de identifi-
2017 cação biométrica: uma abordagem de implementação escalável
/ Caio Arce Nishibe.-- 2017.
64 f. : il. ; 30 cm

Texto em português, com resumo em inglês
Disponível também via World Wide Web
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2017
Bibliografia: f. 61-64

1. Biometria. 2. Identificação biométrica. 3. Computação de alto desempenho. 4. Big data. 5. Antropometria. 6.Impressões digitais. 7. Sistemas de memória de computadores. 8. Engenharia elétrica - Dissertações I. Fonseca, Keiko Verônica Ono, orient. II. Universidade Tecnológica Federal do Paraná - Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: Ed. 22 – 621.3

Biblioteca Central da UTFPR, Câmpus Curitiba
Bibliotecária : Anna T. R. Caruso CRB9/935

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO DE DISSERTAÇÃO Nº 773

A Dissertação de Mestrado intitulada **"Central de Confrontos para um Sistema Automático de Identificação Biométrica: Uma Abordagem de Implementação Escalável"** defendida em sessão pública pelo(a) candidato(a) **Caio Arce Nishibe**, no dia 19 de outubro de 2017, foi julgada para a obtenção do título de Mestre em Ciências, área de concentração Telecomunicações e Redes, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial.

BANCA EXAMINADORA:

Prof(a). Dr(a). Keiko Verônica Ono Fonseca - Presidente – (UTFPR)

Prof(a). Dr(a). Andrey Elisio Monteiro Brito - (UFCG)

Prof(a). Dr(a). Luiz Celso Gomes Junior - (UTFPR)

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 19 de outubro de 2017.

Dedico este trabalho aos amigos, colegas de trabalho e familiares.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela força espiritual para a realização deste trabalho.

À toda comunidade da Universidade Tecnológica Federal do Paraná (UTFPR) pelo apoio novamente recebido em mais uma fase de aprendizado.

À Professora Dr[a]. Keiko Verônica Ono Fonseca pela orientação deste trabalho e pelos momentos de aprendizado.

Ao Dr. Maurizio Tazza, ao MSc Alvaro Cardoso de Matos Júnior e ao MSc André Zanin Rovani pelos momentos de aprendizado e principalmente por permitir o desenvolvimento deste trabalho em conjunto com as atividades de sua empresa.

A todos os funcionários da Antheus Tecnologia, que sempre me apoiaram e me deram forças para continuar e nunca desistir.

Por último, mas não menos importante, agradeço à minha família e principalmente à minha namorada Mari, pois sem o apoio dela, certamente não conseguiria vencer mais esse desafio.

"It matters not what someone is born, but what they grow to be."

– J.K. Rowling, Harry Potter and the Goblet of Fire

RESUMO

Nishibe, Caio Arce. CENTRAL DE CONFRONTOS PARA UM SISTEMA AUTOMÁTICO DE IDENTIFICAÇÃO BIOMÉTRICA: UMA ABORDAGEM DE IMPLEMENTAÇÃO ESCALÁVEL. 64 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Com a popularização do uso da biometria, determinar a identidade de um indivíduo é uma atividade cada vez mais comum em diversos contextos: controle de acesso físico e lógico, controle de fronteiras, identificações criminais e forenses, pagamentos. Sendo assim, existe uma demanda crescente por Sistemas Automáticos de Identificação Biométrica (ABIS) cada vez mais rápidos, com elevada acurácia e que possam operar com um grande volume de dados. Este trabalho apresenta uma abordagem de implementação de uma central de confrontos para um ABIS de grande escala utilizando um *framework* de computação em memória. Foram realizados experimentos em uma base de dados real com mais de 50 milhões de impressões digitais em um *cluster* com até 16 nós. Os resultados mostraram a escalabilidade da solução proposta e a capacidade de operar em grandes bases de dados.

Palavras-chave: ABIS, biometria, confronto biométrico, Big Data, HPC.

ABSTRACT

Nishibe, Caio Arce. MATCHING PLATFORM FOR AN AUTOMATIC BIOMETRIC IDENTIFICATION SYSTEM: A SCALABLE IMPLEMENTATION APPROACH. 64 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

With the popularization of biometrics, personal identification is an increasingly common activity in several contexts: physical and logical access control, border control, criminal and forensic identification, payments. Thus, there is a growing demand for faster and accurate Automatic Biometric Identification Systems (ABIS) capable to handle a large volume of biometric data. This work presents an approach to implement a scalable cluster-based matching platform for a large-scale ABIS using an in-memory computing framework. We have conducted some experiments that involved a database with more than 50 million captured fingerprints, in a cluster up to 16 nodes. The results have shown the scalability of the proposed solution and the capability to handle a large biometric database.

Keywords: ABIS, biometrics, match server, Big Data, HPC.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ABIS	*Automated Biometric Identification System*
ACID	*Atomicity, Consistency, Isolation, Durability*
AFIS	*Automated Fingerprint Identification System*
AMQP	*Advanced Message Queuing Protocol*
ANSI	*American National Standards Institute*
API	*Application Programming Interface*
CPU	*Central Processing Unit*
DML	*Data Manipulation Language*
DNA	*Deoxyribonucleic Acid*
FAR	*False Aceptance Rate*
fgp	*finger position*
FIFO	*First In, First Out*
FRR	*False Rejection Rate*
GC	*Garbage Collection*
Gbps	*Gigabits per second*
HDFS	*Hadoop Distributed File System*
HPC	*High Performance Computing*
IMDG	*In-Memory Data Grid*
JDBC	*Java Database Connectivity*
JPA	*Java Persistence API*
JTA	*Java Transaction API*
JTS	*Java Transaction Service*
LAN	*Local Area Network*
MB	*Megabyte*

MOM	*Message Oriented Middleware*
MPI	*Message Passing Interface*
mutex	*mutual exclusion*
NGI	*Next Generation Identification*
OpenMP	*Open Multi-Processing*
ppi	*pixels per inch*
RAM	*Random Access Memory*
RDD	*Resilient Distributed Dataset*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
SQL	*Structured Query Language*
UIDAI	*Unique Authority of India*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo contextualiza o cenário em que este trabalho está inserido, além de definir o problema que foi proposta uma solução. Discute também a relevância deste trabalho, bem como apresenta a estrutura das demais partes deste documento.

1.1 CONTEXTUALIZAÇÃO

Reconhecimento biométrico é o processo de estabelecer, de forma automática, a identidade de um indivíduo através de características físicas, químicas e comportamentais (National Science and Technology Council, 2006). Estas características, também chamadas de identificadores biométricos ou traços biométricos, podem ser: impressão digital, impressão palmar, face, íris, voz, geometria da mão, formato da orelha, forma de caminhar, padrão vascular, DNA, entre outras. (JAIN; FLYNN; ROSS, 2008).

A identidade de um indivíduo precisa ser verificada ou determinada diariamente em diversos contextos, como por exemplo, para controle de acesso físico e lógico, controle de presença, solução de crimes, identificação de cadáveres, controle de recebimento de benefícios, autenticação de eleitores e autenticação de transações financeiras. Utilizar identificadores biométricos para reconhecer um indivíduo é uma forma mais conveniente e robusta do que através de métodos que empregam somente *tokens* e/ou senhas. Isto se deve ao fato de que os traços biométricos não podem ser facilmente forjados, extraviados ou compartilhados (MALTONI et al., 2009). Além disso, a biometria é um excelente mecanismo para autenticar ou identificar um indivíduo devido as suas propriedades de unicidade, estabilidade e persistência (RATHA; CONNELL; PANKANTI, 2015).

Nos últimos anos, o uso do reconhecimento biométrico tem aumentado e se tornado um tópico importante para muitas instituições e empresas. Com isso, as bases de dados biométricos estão se tornando cada vez maiores, podendo conter milhões e até dezenas de milhões de registros biométricos multimodais (PERALTA et al., 2016; RATHA; CONNELL; PANKANTI, 2015).

1.2 PROBLEMA INVESTIGADO E RELEVÂNCIA

Este trabalho tem como objetivo principal apresentar uma abordagem para implementação de uma central de confrontos para um ABIS que seja capaz de operar com dezenas de milhares de registros multibiométricos. Para tal, é proposta a utilização do Apache Ignite como base desta implementação, permitindo que se obtenha escalabilidade, processamento distribuído e altamente paralelizável e desempenho suficientes para atender as exigências do mercado.

1.3 ESTRUTURA DO DOCUMENTO

O Capítulo 2 apresenta a fundamentação teórica contemplando uma revisão literária sobre sistemas biométricos, seus modos de operação e uma breve descrição das principais soluções *open source* em IMDG. No Capítulo 3, são apresentados os trabalhos relacionados que serviram de base na análise e justificativa deste trabalho. O Capítulo 4 formaliza e detalhada a abordagem de implementação proposta. No Capítulo 5, são descritos os experimentos realizados e a avaliação dos resultados obtidos. O Capítulo 6 contém as considerações finais e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma breve introdução aos conceitos necessários para o desenvolvimento deste trabalho. Alguns fatos históricos são descritos, bem como a definição de sistemas biométricos e seus modos de operação. É apresentado o paradigma de computação em memória e os principais *frameworks open source*.

2.1 FATOS HISTÓRICOS

Em 1858, Sir William Herschel, um magistrado britânico radicado na Índia, começou a utilizar biometria da forma como é entendida atualmente. Herschel adotou a prática de autenticar os contratos dos trabalhadores de seu distrito através de uma assinatura biométrica, além da assinatura convencional. Inicialmente, o magistrado coletava a impressão completa da palma da mão, porém com o passar do tempo, ele começou a coletar apenas a impressão digital de dois dedos (KOMARINSKI, 2005).

Em 1880, o cirurgião britânico Dr. Henry Faulds publicou um estudo sobre o uso de impressões digitais como forma de identificação. Faulds encaminhou seu trabalho para seu sobrinho, Sir Francis Galton, que identificou as características básicas das impressões digitais: as minúcias. (CAMPISI et al., 2010a).

Com a industrialização e o rápido crescimento populacional, verificar a identidade de um indivíduo se tornou cada vez mais complicado. Alphonse Bertillon acreditava que o sistema de identificação criminal empregado na época era inviável, pois se baseava somente em testemunhas oculares (CAMPISI et al., 2010a). Em 1883, o francês Bertillon criou um sistema novo, chamado de antropometria. Tal sistema era baseado nas medidas do corpo e registros fotográficos. Bertillon armazenava fichas (Figura 1) contendo a cor dos olhos, o comprimento dos dedos mínimo e médio, o comprimento e a largura da cabeça, o comprimento do pé esquerdo, o comprimento do antebraço medido a partir do cotovelo até a ponta do dedo médio, entre outras medidas (KOMARINSKI, 2005).

Na década de 1890, Sir Edward Henry, inspetor geral da polícia de Bengala, queria um sistema que substituísse a antropometria. Henry consultou Galton e implantou seus métodos em seu departamento. Juntamente com os oficiais Khan Bahadur Azizul Haque e Rai Bahaden Hem Chandra Bose, Henry criou um sistema fácil e eficiente de classificação, armazenamento e busca de impressões digitais (KOMARINSKI, 2005). Este método, conhecido como sistema de classificação Henry, foi posteriormente adotado pelo FBI e ainda é utilizado em alguns países (CAMPISI et al., 2010a).

No ano de 1903, a antropometria parou de ser utilizada na maioria das jurisdições norte americanas. Isto ocorreu pois, medidas idênticas e fotografias de baixa qualidade foram atribuídas

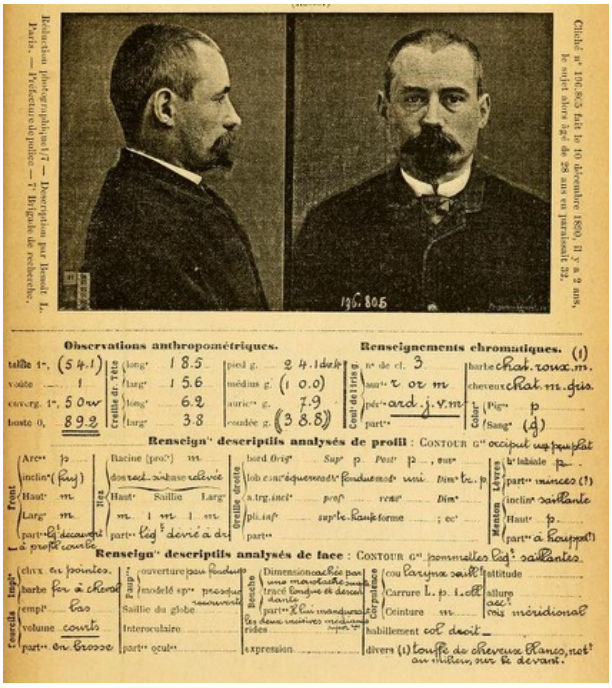

Figura 1 – Exemplo de ficha antropométrica.
Fonte: Bertillon (1893)

a dois prisioneiros na penitenciária federal em *Fort Leavenworth*, no Kansas. Para solucionar este impasse, as impressões digitais foram utilizadas e seu uso se disseminou pelo sistema criminal de justiça dos Estados Unidos (CAMPISI et al., 2010a).

Em 1946, o FBI já possuía em seus arquivos mais de 100 milhões de fichas decatatilares. No ano de 1971, este número já tinha ultrapassado os 200 milhões, fazendo com que o armazenamento e a conservação de tais fichas se tornasse um problema (KOMARINSKI, 2005).

2.2 CARACTERÍSTICAS DA IMPRESSÃO DIGITAL

Os padrões presentes nas impressões digitais podem ser analisados em diferentes escalas. Em um nível global (características de nível 1), o fluxo das linhas forma estruturas que são utilizadas para classificar e indexar impressões digitais. As singularidades, que podem ser núcleos ou deltas, atuam como pontos de controle onde as linhas se dobram. Uma impressão digitial pode ser classificada quanto à sua geometria (posicionamento dos núcleos e deltas) em três principais grupos: arcos, presilhas e verticilos (MALTONI et al., 2009).

Um arco plano (Figura 2a) possui cristas papilares que entram por um lado, se elevam em uma pequena protuberância, e saem pelo lado oposto ao que entraram. Arcos deste tipo não possuem núcleos ou deltas. Os arcos tendos (Figura 2b), que são similares aos arcos planos, possuem ao menos uma crista com alta curvatura e contém um núcleo e um delta. Uma presilha possui uma ou mais cristas papilares que entram por um lado, dão a volta formando um laço e saem pelo mesmo lado que entraram. Presilhas que possuem o delta na direita do observador são chamadas de presilhas internas (Figura 2c). Já as que possuem o delta na esquerda do observador

(a) (b) (c) (d) (e)

Figura 2 – Classificação das impressões digitais: (a) arco plano, (b) arco tendo, (c) presilha interna, (d) presilha externa, (e) verticilo. Núcleos são marcados com um círculo e deltas com um triângulo. Fonte: Autoria própria.

são chamadas de presilhas externas (Figura 2d). Um verticilo (Figura 2e) contém ao menos uma crista papilar que faz uma volta completa ao redor do centro da impressão digital. Possuem dois núcleos e dois deltas (MALTONI et al., 2009).

Em um nível local (características de nível 2), é possível observar as minúcias, que são pontos característicos de uma crista papilar. As minúcias (Figura 3) são estáveis e robustas ao processo de impressão da digital e podem ser de dois tipos: fim de linha e bifurcação. O fim de linha (Figura 3a) é o ponto onde a crista papilar termina abruptamente. A bifurcação (Figura 3b) é o ponto onde a crista papilar se divide em dois ramos (MALTONI et al., 2009).

(a) (b)

Figura 3 – Tipos de minúcias: (a) fim de linha, (b) bifurcação. Fonte: Autoria própria.

Em um nível mais detalhado (características de nível 3), detalhes intra-crista podem ser detectados (somente com resolução maior ou igual a 1000 ppi). Estas incluem: largura, formato, curvatura, contorno da crista papilar e também pontos e cristas incipientes. As características de nível 3 mais importantes são os poros (Figura 4), cuja posição e forma são altamente discriminativos (MALTONI et al., 2009).

2.3 SISTEMAS BIOMÉTRICOS

Um sistema biométrico é um sistema de reconhecimento de padrões capaz de capturar dados biométricos de um indivíduo, extrair as características relevantes (*features*), comparar esse conjunto de características, também chamado de *template* biométrico, contra um ou mais conjuntos armazenados em uma base de dados, e por fim, executar uma ação baseada no resultado

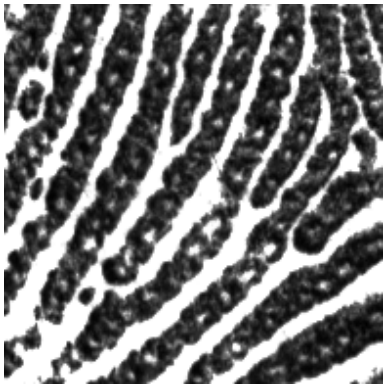

Figura 4 – Poros somente podem ser extraídos em imagens de alta resolução.
Fonte: Autoria própria.

desta comparação. Neste tipo de sistema, podem ser identificados quatro módulos principais: (a) um módulo de captura; (b) um módulo de controle de qualidade e extração de características; (c) um módulo de confronto; (d) um módulo de armazenamento (JAIN; FLYNN; ROSS, 2008).

O módulo de captura requer um leitor ou escâner biométrico adequado que capture os traços biométricos brutos de um indivíduo. No módulo de controle de qualidade e extração de características a qualidade desses dados é avaliada. Se estiver fora dos padrões de operação do sistema, tais dados podem ser rejeitados, sendo necessária a apresentação de outra amostra biométrica. Caso a qualidade esteja dentro dos padrões aceitáveis pelo sistema, a amostra é processada e um *template* biométrico é gerado. O módulo de confronto compara este *template*, chamado de *template* de prova, contra os demais *templates* armazenados, chamados de galeria ou referência, e gera escores de similaridade. Estes escores são utilizados para validar a identidade de um indivíduo ou gerar uma lista de possíveis candidatos. O módulo de armazenamento é um repositório de dados biométricos (JAIN; FLYNN; ROSS, 2008).

Sistemas biométricos podem operar em dois modos distintos: verificação e identificação (JAIN; FLYNN; ROSS, 2008).

2.3.1 Verificação (1:1)

A verificação, também chamada de autenticação, é o modo mais simples de operação, onde apenas uma identidade reivindicada deve ser confirmada através de uma ou mais capturas biométricas (RATHA; CONNELL; PANKANTI, 2015). Seja A_p uma amostra biométrica de prova e A_r uma amostra biométrica de referência. O processo de verificação traduz-se em determinar se duas amostras biométricas A_p e A_r pertencem ao mesmo indivíduo realizando uma comparação 1:1 (JAIN; HONG; BOLLE, 1997). A resposta do sistema (Figura 5) pode ser a aceitação (genuíno) ou rejeição (impostor) da identidade reivindicada dependendo do escore de similaridade entre as amostras biométricas (PERALTA et al., 2014).

Sistemas biométricos de verificação podem ser agrupados nas seguintes categorias (CAMPISI et al., 2010c):

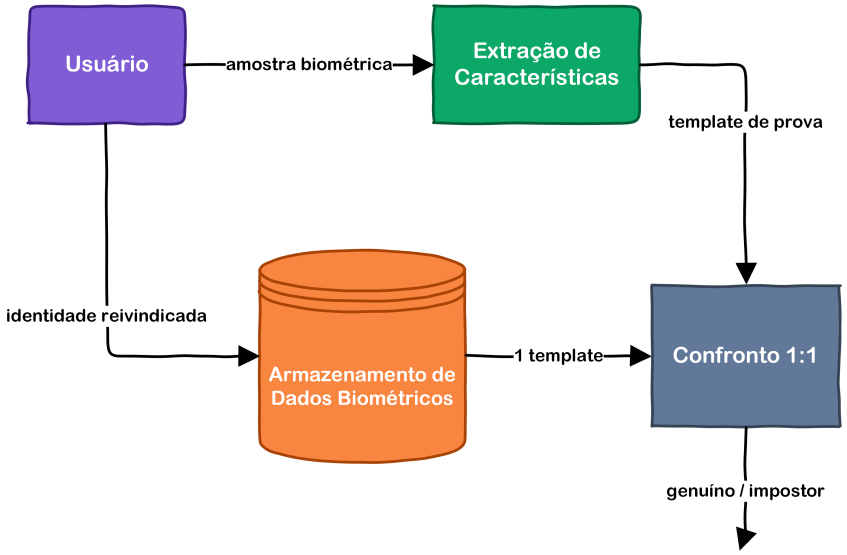

Figura 5 – Processo de verificação, também chamado de autenticação ou confronto 1:1. Neste modo de operação o sistema valida a identidade reivindicada através da comparação entre os dados biométricos capturados ao vivo e os *templates* deste indivíduo previamente armazenados.
Fonte: Adaptado de Jain, Flynn e Ross (2008)

- Sistemas de controle de acesso lógico: aplicações de autenticação por voz em comércio eletrônico; cartões bancários com *templates* biométricos para complementar o uso de senhas e demais dados bancários; sistemas de escaneamento do padrão vascular em caixas eletrônicos; autenticação em computadores, celulares, *softwares* e redes através de impressão digital.

- Sistemas de controle de acesso físico: controle carcerário de prisioneiros e visitantes para evitar a troca de identidades; sistemas de reconhecimento facial, íris e impressões digitais que verificam a identidade de viajantes e auxiliam no controle de fronteiras; controle de acesso a usinas nucleares e outras instalações de acesso restrito.

- Sistemas de obtenção de direitos e privilégios: autenticação para recebimento de benefícios do governo; autenticação de motoristas; autenticação de eleitores.

2.3.2 Identificação (1:N)

A identificação é um processo extremamente complexo que envolve operações de busca em grandes bases de dados biométricos, geralmente incluindo alguns dados biográficos e outros metadados, mantendo as taxas de erro extremamente baixas (RATHA; CONNELL; PANKANTI, 2015). Seja $T_r = \{t_1, t_2, ..., t_n\}$ o conjunto contendo n *templates* utilizados como referência em uma identificação. O *template* de prova T_p, gerado a partir da amostra de prova A_p, deverá ser comparado contra todos os *templates* $t_i \in T_r$, realizando n comparações 1:1. A resposta do sistema (Figura 6) pode ser não identificado (não encontrado) ou uma lista de candidatos (PERALTA et al., 2014).

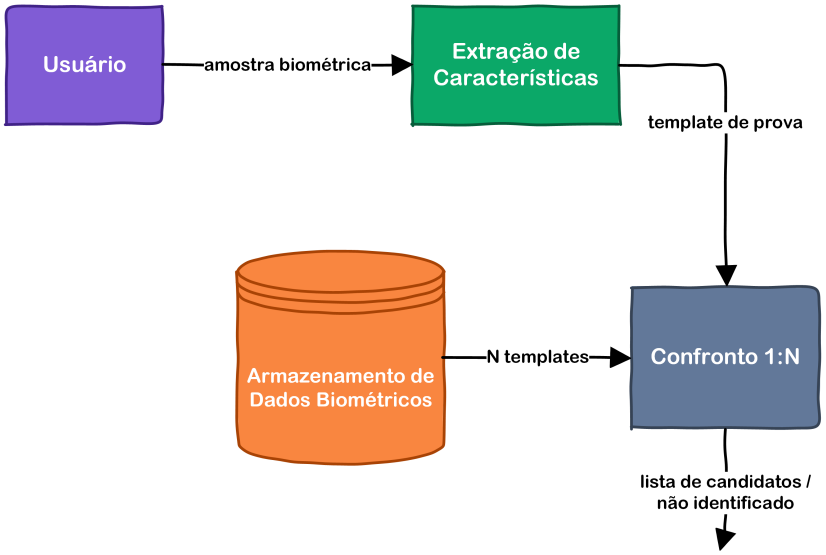

Figura 6 – Processo de identificação, também chamado de confronto 1:N. Neste modo de operação, o sistema estabelece a identidade de um indivíduo através da comparação dos dados biométricos capturados contra todos os *templates* armazenados no banco de dados biométrico.
Fonte: Adaptado de Jain, Flynn e Ross (2008)

Sistemas biométricos de identificação podem ser divididos nas seguintes categorias (CAMPISI et al., 2010c):

- Sistemas de identificação positiva: comparam uma amostra biométrica não identificada contra todos os registros biométricos armazenados para auxiliar na descoberta da identidade do indivíduo que forneceu a amostra. Exemplos de tais sistemas são: identificação de criminosos através de evidências biométricas coletadas em cenas de crimes; identificação de vítimas; identificação de crianças desaparecidas.

- Sistemas de identificação negativa: comparam uma amostra biométrica identificada contra todos os registros biométricos armazenados para garantir que não exista um indivíduo já cadastrado no sistema com os mesmos dados biométricos submetidos ao confronto. Exemplos de tais sistemas são: emissão de documentos (carteira de identidade, carteira nacional de habilitação, passaporte, título eleitoral) onde apenas um único documento deve ser emitido por indivíduo; identificação de suspeitos, criminosos e terroristas em aeroportos ou multidões (*watchlist*).

2.4 SISTEMAS BIOMÉTRICOS E *BIG DATA*

Ambos os modos de operação de um sistema biométrico (1:1 e 1:N) podem ser considerados exemplos de sistemas conhecidos como de Big Data, isto é, de grande volume de dados. Sendo assim, podem ser identificadas quatro dimensões críticas associadas a tais sistemas: volume, velocidade, variedade e veracidade (RATHA; CONNELL; PANKANTI, 2015).

2.4.1 Volume

Em alguns sistemas biométricos modernos existe uma quantidade grande de dados. Um exemplo disso é o FBI NGI, um repositório de dados biométricos e antecedentes criminais que possui mais de 114 milhões de registros biométricos e é capaz de realizar pesquisas de impressões digitais em uma média de 72 minutos (FBI, 2016).

Outro exemplo é o Aadhaar, maior projeto de identificação biométrica do mundo, criado pelo governo indiano no ano de 2009 em parceria com a UIDAI. Este programa de recenseamento iniciou em 2010 e em abril de 2016 atingiu a marca de 1 bilhão de números Aadhaar emitidos. Atualmente são gerados em média 1 milhão de cadastros por dia. Até o final de 2017 serão 1,2 bilhões de pessoas cadastradas (Safran Identity & Security, 2016). Cada registro possui o tamanho de 8 MB. Sendo assim, atualmente são mais 8 petabytes de dados brutos e 25 petabytes em dados replicados (KRICHEN, 2016).

Trabalhar com um volume de dados desta dimensão é um grande desafio para os sistemas biométricos, principalmente durante a operação de identificação (RATHA; CONNELL; PANKANTI, 2015).

2.4.2 Velocidade

A velocidade com que os dados chegam também é preocupante, principalmente em grandes sistemas civis e corporativos, como por exemplo, comércio eletrônico e sistemas bancários, que utilizam biometria para autenticar todas as suas milhares de transações diárias (RATHA; CONNELL; PANKANTI, 2015).

Para garantir a unicidade de cada Aadhaar emitido e evitar redundâncias e fraudes, são realizadas mais de 1 milhão de operações 1:N por dia em um *cluster* composto por 800 servidores, com tempo de resposta menor que 10 segundos e acurácia de 99,9%. No módulo de autenticação são realizadas 2.800 operações de verificação por segundo utilizando íris e impressões digitais, chegando ao pico de 100 milhões de verificações em 10 horas (KRICHEN, 2016).

2.4.3 Variedade

Sistemas biométricos modernos têm sido implementados para operar com uma grande variedade de tipos de dados: imagens bidimensionais em escala de cinza (impressão digital e palmar), imagens coloridas e em espectro não visível (face e íris), imagens tridimensionais (face), vídeo (face e forma de caminhar) e sinais temporais unidimensionais (voz e assinatura) (RATHA; CONNELL; PANKANTI, 2015).

Os registros biométricos do NGI podem conter impressão digital e palmar, face e íris (FBI, 2016). Já no Aadhaar, durante o cadastro são coletados dados biográficos (nome, endereço, sexo e data de nascimento) e dados biométricos (impressões digitais, íris e uma fotografia frontal) (Safran Identity & Security, 2016).

Além disso, existem tipos diferentes de *templates* para cada tipo de biometria e em algumas situações, diferentes formatos para uma mesma biometria, e múltiplos algoritmos de confronto (RATHA; CONNELL; PANKANTI, 2015).

2.4.4 Veracidade

A dimensão da veracidade se preocupa com o gerenciamento de taxas de erro intrínsecas dos sistemas biométricos, principalmente a relação entre falso-positivos (FAR) e falso-negativos (FRR) (RATHA; CONNELL; PANKANTI, 2015). O desafio de operar em grandes bases de dados deve-se ao fato de que os níveis de acurácia do sistema diminuem significativamente com o aumento da base de dados ou com o aumento do número de usuários realizando operações de identificação (CAMPISI et al., 2010b).

Algumas formas de reduzir esses erros são: detecção de tentativa de fraude (*spoofing*), uso de dados biométricos de boa qualidade (RATHA; CONNELL; PANKANTI, 2015), uso de algoritmo e/ou biometria que possua a menor sobreposição das curvas de FAR e FRR, alterar o limiar para que FAR seja menos provável de acontecer e utilizar fusão de biometrias para reduzir o FRR (CAMPISI et al., 2010b).

2.5 COMPUTAÇÃO EM MEMÓRIA

Com o aumento da capacidade de coletar dados, a demanda por análise e processamento destes dados também aumentou. Desta forma, a computação de alto desempenho (HPC), está se tornando cada vez mais popular. Isto se deve ao fato de que a HPC deixou de ser empregada apenas na modelagem de fenômenos físicos complexos, e passou a ser parte integrante da indústria (BHUIYAN; ZHELUDKOV; ISACHENKO, 2016).

Suponha que seja preciso processar 1 milhão de transações bancárias por segundo. Utilizando a computação tradicional baseada em disco, é pouco provável que tal performance seja obtida. Nestes casos é preciso utilizar outro paradigma, a computação em memória, que mantém os dados na memória RAM do servidor para que o processamento ocorra ordens de magnitude mais rápido do que com tecnologias baseadas somente em disco. (BHUIYAN; ZHELUDKOV; ISACHENKO, 2016).

2.5.1 *In-Memory Data Grid*

De forma simplificada, um *data grid* pode ser visto como um *cache* distribuído onde os dados são replicados em memória, para que fiquem acessíveis de qualquer ponto do *cluster*. Em geral, *data grids* fazem isso particionando dados em memória, onde cada membro do *cluster* fica responsável por uma porção dos dados. Sendo assim, quanto maior o número de nós no *cluster*, mais dados podem ser armazenados (SETRAKYAN, 2015).

Além de cache em memória, *data grids* possuem outras funcionalidades: transações distribuídas, consultas distribuídas, suporte a SQL e integração com banco de dados (SETRAKYAN, 2015).

As principais soluções *open source* em IMDG são: Hazelcast IMDG (Hazelcast, 2017), Infinispan (Red Hat, 2017b) e Apache Ignite (The Apache Software Foundation, 2017b). Elas oferecem um conjunto similar de funções básicas, apresentando pequenas diferenças no uso, performance e funcionalidades adicionais.

2.5.2 Hazelcast IMDG

O Hazelcast IMDG é uma plataforma distribuída para IMDG que permite escalabilidade horizontal de processamento e armazenamento, além de fornecer um conjunto de estruturas de dados e funcionalidades distribuídas (JOHNS, 2013). Pode ser implantado em servidores locais (físicos ou virtualizados), em contêineres Docker (Docker, 2017) e na nuvem. Todos os membros do *cluster* possuem responsabilidades iguais, não existindo a noção de mestre-escravo, evitando assim a criação de um ponto único de falha (LUCK, 2017).

As principais funcionalidades do Hazelcast IMDG são (Hazelcast, 2017):

- **Estruturas de dados distribuídas**: os dados podem ser armazenados em vários tipos de estruturas de dados distribuídas (lista de pares chave-valor, coleção de objetos, coleção de objetos únicos, fila FIFO), além de permitir o envio de mensagens entre os membros do *cluster*, utilizando um modelo publicar-assinar, e a sincronização de execução concorrente através de *mutex*.
- **Computação distribuída**: permite a execução de tarefas distribuídas e o processamento de dados do tipo chave-valor.
- **Consulta distribuída**: permite a execução de agregações e consultas, inclusive utilizando SQL.
- **Armazenamento**: as estruturas de dados são armazenadas serializadas no espaço de pilha (*heap*) do Java e estão sujeitas à GC. Para evitar problemas com longas pausas de GC e travamentos do *cluster*, existe um *plug-in* somente na versão *Enterprise HD*, que não é *open-source*.

2.5.3 Infinispan

O Infinispan é uma plataforma distribuída de armazenamento em memória de dados do tipo chave-valor, escrita em Java. Pode ser utilizado como uma biblioteca ou através de um serviço acessado remotamente utilizando diversos protocolos, como: REST, *WebSockets*, *Memcached* (Red Hat, 2017a). A proposta deste IMDG é prover estruturas de dados distribuídas

e concorrentes, geralmente utilizadas para realizar *cache* distribuído, armazenamento de dados chave-valor ou objetos de banco de dados (Red Hat, 2017b).

As principais funcionalidades que o Infinispan fornece são (Red Hat, 2017a):

- **Cache local e distribuído**: armazenamento no *heap* ou fora dele de referências de objetos ou representações binárias compactas.
- **Persistência**: possibilidade de utilizar um meio de armazenamento externo (arquivo, JDBC, Infinispan Remote Server, JPA).
- **Transações distribuídas**: controle de concorrência baseado em *locks* otimistas e pessimistas e ordem total. Integração com JTA/JTS.
- **Indexação e consulta**: consultas indexadas e não-indexadas, agregação e agrupamento.
- **Execução de código**: execução distribuída de tarefas e *scripts* utilizando *streams*.

2.5.4 Apache Ignite

O Apache Ignite é uma plataforma de computação em memória, implementado em Java, que inclui uma coleção de componentes independentes e bem integrados, projetados para trabalhar em tempo real com grande volume de dados, aumentando a performance e a escalabilidade de uma determinada aplicação. A principal diferença do Ignite para os demais *frameworks* é o maior número de funcionalidades e a simplicidade de uso de sua API. Além disso, é o único que disponibiliza um conjunto de componentes chamado de *Hadoop Accelerator* e *Ignite Shared RDD* que permitem o processamento em tempo real para usuários de Hadoop e Spark (BHUIYAN; ZHELUDKOV; ISACHENKO, 2016).

O modo com que o Apache Ignite foi projetado, faz com que ele seja por si só um sistema altamente disponível e escalável horizontalmente. A comunicação entre nós ponto-a-ponto permite que todos recebam rapidamente atualizações sem a necessidade de um nó mestre. Proporciona ainda a inclusão ou remoção de nós sem afetar a operação do sistema e possui detecção automática de falhas e recuperação de um ou mais nós (BHUIYAN; ZHELUDKOV; ISACHENKO, 2016). Permite o armazenamento de cópias dos dados (*backup*), fazendo com que o sistema seja tolerante a falhas parciais do *cluster*. Quando a persistência em disco está habilitada, os dados armazenados no sistema se tornam tolerantes a falhas totais do *cluster* (GridGrain Systems, 2017).

As principais funcionalidades do Apache Ignite são (GridGrain Systems, 2017):

- **Memória durável**: a arquitetura de memória durável (Figura 7), apresentada recentemente, permite o armazenamento e o processamento de dados e índices em memória e em disco, aliando desempenho da memória RAM e a durabilidade do disco. Mantém o conjunto

completo de dados e índices no disco. Esta funcionalidade permite reiniciar o *cluster* rapidamente, sem a necessidade de carregar dados de uma fonte externa.

- **Banco de dados SQL distribuído**: escalável horizontalmente, tolerante a falhas e compatível com SQL ANSI-99. Possui ainda suporte a todos os comandos DML.
- ***Data grid***: armazenamento em memória com esquema chave-valor. Suporta transações distribuídas ACID, tratamento de exceções, balanceamento avançado de carga, consultas compatíveis com SQL ANSI-99, *cache* de nível 2 para Hibernate (*Hibernate L2 Cache*), persistência automática e integração com bases de dados externas.
- ***Compute grid***: permite a execução de tarefas de forma distribuída e paralela. Apresenta escalabilidade linear e baixa latência. Além disso possui tolerância a falhas, balanceamento de carga e controle de execução por diversos critérios (prioridade e fila).
- **Mensagens e eventos distribuídos**: provê um sistema de troca de mensagens entre os membros do *cluster* utilizando os modelos publicar-assinar e ponto-a-ponto. Através dos eventos distribuídos, é possível receber notificações sobre diversos tipos de eventos que acontecem no *cluster* como, por exemplo, execução de tarefas, operações de escrita, leitura e consulta dos dados.
- ***Hadoop Accelerator***: inclui um sistema de arquivos em memória compatível com o HDFS e uma implementação otimizada do *MapReduce* (DEAN; GHEMAWAT, 2008) em memória, permitindo um ganho de desempenho de até 100 vezes.
- ***Ignite Shared RDD***: fornece uma visão compartilhada em memória dos dados para diferentes tarefas Spark, *workers* e aplicações, enquanto na implementação nativa do *Spark* RDD, os dados não podem ser compartilhados entre tais tarefas ou aplicações e são imutáveis.

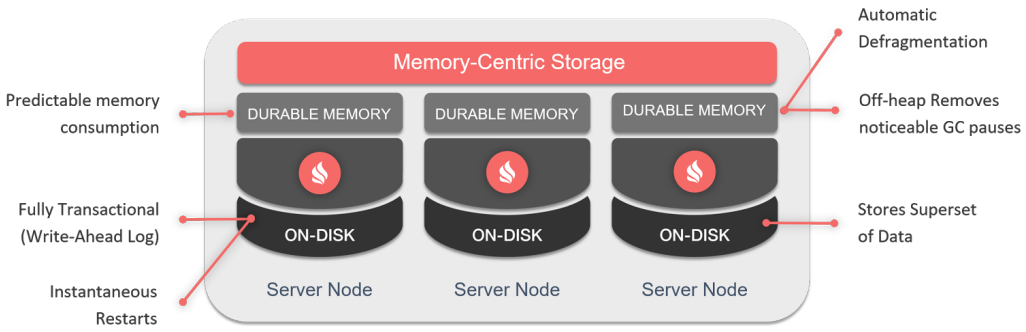

Figura 7 – Arquitetura de memória durável do Apache Ignite.
Fonte: GridGrain Systems (2017)

Neste trabalho foi utilizado o Apache Ignite pela facilidade no uso de sua API Java, pelos resultados nos *benchmarks* (GridGain Systems, 2017) e pelas funcionalidades adicionais que são adequadas ao contexto da solução proposta. A memória durável é uma funcionalidade

desejável, visto que, em caso de necessidade de reiniciar o *cluster*, não é necessário aguardar a carga dos *templates* dos *templates* para a memória. Além disso, através de *queries* distribuídas é possível garantir a varredura de todos os registros biométricos, mesmo durante mudanças de topologia do *cluster*, através de controle de execução de tarefas distribuídas. Por fim, a facilidade na integração com banco de dados, de forma simples e robusta, é outro fator relevante na escolha do Apache Ignite.

2.6 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou os fundamentos teóricos dos diversos conceitos envolvidos neste trabalho. Iniciou-se com a descrição de alguns fatos históricos importantes para a compreensão do desenvolvimento dos sistemas biométricos. A seguir, o conceito de sistema biométrico e seus modos de operação foram abordados. Em seguida, as características das impressões digitais foram apresentadas. Além disso, foram abordados os principais desafios referentes aos sistemas biométricos de grande escala. Por fim, o paradigma de computação em memória e os principais *frameworks open source* foram descritos.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta um levantamento dos trabalhos relacionados. A noção de Sistema Automático de Identificação Biométrica (ABIS) é recente e a literatura especializada não contém trabalhos específicos. Entretanto, o ABIS é uma generalização do Sistema Automático de Identificação de Impressões Digitais (AFIS), logo os conceitos referentes ao AFIS podem ser aplicados ao ABIS com pequenas modificações. Por fim, algumas considerações são feitas sobre como a solução proposta utiliza os conhecimentos aprendidos e quais são os diferenciais em relação aos trabalhos existentes.

3.1 IDENTIFICAÇÃO BIOMÉTRICA EM GRANDES BASES DE DADOS

De acordo com Peralta et al. (2014), para que um sistema biométrico seja capaz de operar com grandes bases de dados, alguns requisitos devem ser ponderados:

- **Precisão**: as taxas de erro intrínsecas (FAR e FRR) devem ser reduzidas o máximo possível.
- **Eficiência**: está relacionada com o tempo gasto pelo sistema para encontrar uma correspondência na base de dados. Este tempo de resposta depende das especificidades de cada sistema e do contexto de uso, mas deve ser o menor possível.
- **Escalabilidade**: mostra a capacidade do sistema operar com diferentes tamanhos de bases de dados biométricos, mantendo os requisitos de precisão e eficiência através do aumento ou redução dos recursos de *hardware*.
- **Flexibilidade**: o sistema deve ser capaz de operar em qualquer tamanho de base de dados e em diferentes configurações de *hardware* (arquiteturas, número de computadores, processadores e sistemas operacionais).

Bey et al. (2008) descrevem uma arquitetura para implementação de um AFIS distribuído, baseada no modelo mestre-escravo, utilizando agentes (Figura 8). O diferencial deste trabalho é utilizar o poder de processamento ocioso dos computadores dos usuários do sistema, conectados em uma LAN, para realizar os confrontos biométricos.

Existem três tipos de agentes no sistema proposto (BEY et al., 2008):

- **Agente de observação**: é instalado em todas as máquinas dos usuários e detecta o estado de utilização da CPU e memória RAM em intervalos de tempos regulares. Essas informações são analisadas, um valor médio é calculado e enviado para o agente de controle.

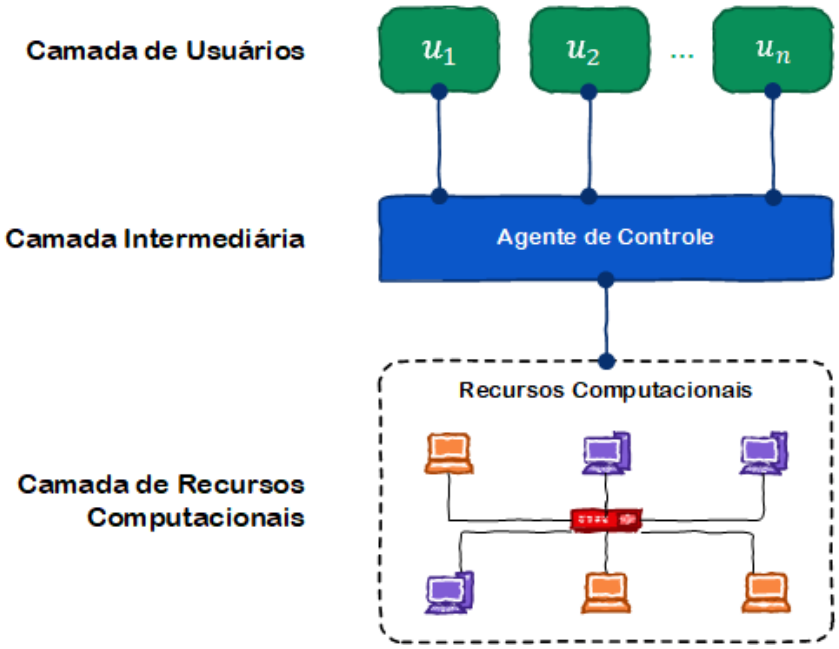

Figura 8 – Arquitetura de AFIS distribuído multiagente proposto por Bey et al. (2008). Seja $U = \{u_1, u_2, ..., u_n\}$ o conjunto de usuários. A camada de usuários permite que cada $u_i \in U$ faça solicitações de confronto e consulte os resultados dessas solicitações. A camada intermediária é responsável por coletar informações sobre os recursos computacionais disponíveis, criar e gerenciar o plano de distribuição e execução das tarefas. A camada de recursos computacionais executa todas as tarefas.

Fonte: Adaptado de Bey et al. (2008)

- **Agente de confronto**: recebe os *templates* de prova e executa uma tarefa de confronto contra um conjunto T_i de *templates* de impressões digitais determinado pelo agente de controle. Ao final, o agente de confronto envia o resultado parcial do confronto para o agente de controle.

- **Agente de controle**: responsável pela coordenação entre as camadas. Recebe os dados de utilização de recursos computacionais, cria um plano de distribuição e execução, seleciona os *hosts* que participarão da execução, distribui os conjuntos T_i de acordo com a capacidade de processamento de cada *host*, gera uma lista final de candidatos e envia para o usuário solicitante.

A maior contribuição do trabalho de Bey et al. (2008) é o algoritmo de balanceamento de carga proposto. Entretanto, foram utilizados apenas 700 registros de impressões digitais para validar a proposta e, o melhor tempo de resposta obtido foi de 3 minutos e 14 segundos, utilizando 20 computadores. Conforme exposto em Ross, Shah e Jain (2007), Feng e Jain (2011) e Cao e Jain (2015), é possível reconstruir a imagem da impressão digital a partir do *template* biométrico. Logo, distribuir os *templates* para os computadores dos usuários sem qualquer proteção é uma vulnerabilidade grave, facilitando o roubo de dados sensíveis e o uso destes dados em fraudes.

Miron e Hulea (2011) apresentam uma arquitetura para implementação de um AFIS distribuído baseada no modelo cliente-servidor. Os clientes são dispositivos móveis (*smartphones*

e *notebooks*) que podem coletar dados biométricos, enviar estes dados para um servidor predeterminado, juntamente com uma requisição (cadastro, verificação ou identificação) e receber os resultados destas. Os servidores processam as solicitações dos clientes e são capazes de encaminhar para outros servidores aquelas não resolvidas localmente. Cada servidor mantém uma partição local da base de dados biométrica. Quando um cliente envia uma solicitação de identificação, por exemplo, para o servidor S_1, este confronta somente com os *templates* armazenados localmente. Se uma correspondência for encontrada, ele retorna para o cliente. Caso contrário, o servidor S_1 repassa a solicitação para S_2, agindo temporariamente como cliente, e assim sucessivamente até que todos os servidores tenham sido consultados ou uma correspondência for encontrada.

São propostas em Miron e Hulea (2011) duas topologias: uma ponto-a-ponto (Figura 9) e uma baseada em MOM (Figura 10). A arquitetura proposta é totalmente inviável para grandes bases de dados, pois os tempos de processamento do AFIS distribuído são maiores do que o AFIS sequencial, quando considera-se um cenário pessimista onde nenhuma correspondência é encontrada em nenhum dos servidores, fazendo com que todos sejam visitados de maneira sequencial. Outro ponto a ser ponderado é que, a base de dados biométricos utilizada nos experimentos continha apenas 200 impressões digitais.

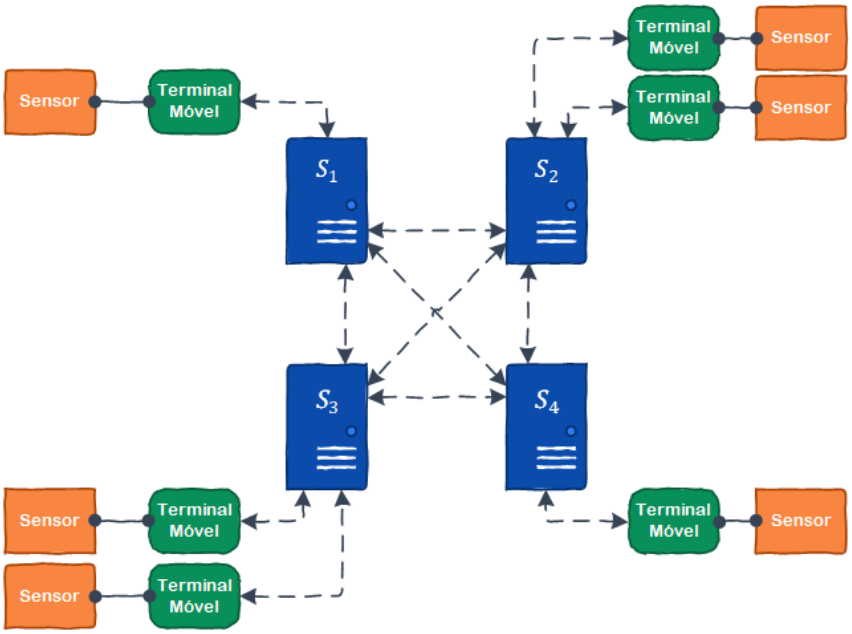

Figura 9 – Topologia ponto-a-ponto utilizada por Miron e Hulea (2011). Seja S o conjunto de servidores definido por $S = \{s_1, s_2, ..., s_n\}$, onde n é o número total de servidores. Cada terminal móvel (cliente) se conecta com somente um servidor $s_i \in S$. O conjunto de conexões do servidor $s_i \in S$ pode ser definido por $X_i = \{x_{ij} \mid j > 0$ e $j \leq n$ e $j \neq i\}$, logo $|X_i| = n - 1$.
Fonte: Adaptado de Miron e Hulea (2011)

Peralta et al. (2014) desenvolveram um *framework* distribuído para realizar identificações (1:N) utilizando impressões digitais, em grandes bases de dados, seguindo o modelo mestre-escravo. Os autores afirmam que a computação de alto desempenho (HPC) é um recurso que pode ser utilizado para reduzir os tempos de identificação, pois o gargalo em um AFIS é o

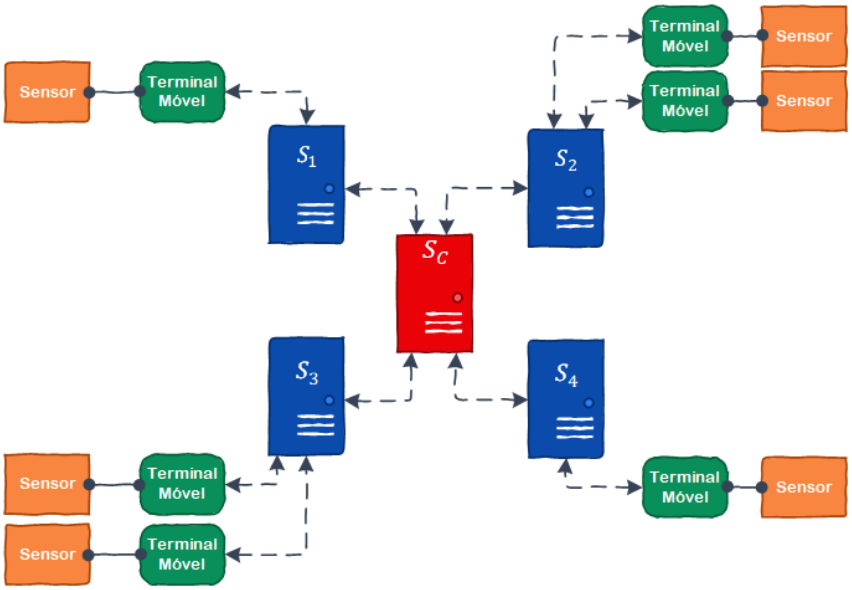

Figura 10 – Topologia baseada em MOM utilizada por Miron e Hulea (2011). Nesta topologia é adicionado um servidor central $s_c \notin S$ com o propósito de reduzir o número de conexões entre os servidores, logo $|C_i| = 1$.

Fonte: Adaptado de Miron e Hulea (2011)

algoritmo de confronto, que é executado de maneira independente n vezes, onde n é o número de *templates* armazenados na base de dados. Enumeram ainda algumas vantagens obtidas pelo uso da HPC para o processamento distribuído e paralelo em um AFIS:

- **Eficiência**: os tempos de resposta podem ser reduzidos através do uso de diversos processadores e computadores para processamento paralelo.
- **Robustez**: o sistema torna-se tolerante a falhas devido ao uso de múltiplos servidores de processamento. Quando uma falha de *hardware* ocorre e uma das máquinas sai de operação, as demais podem assumir a carga desta e mesmo assim produzir respostas corretas.
- **Escalabilidade**: um algoritmo que é capaz de se beneficiar do aumento do poder de processamento pela inclusão de *hardware*, poderá ser utilizado para solucionar problemas de diferentes tamanhos sem a necessidade de modificações.

O *framework* proposto (Figura 11) por Peralta et al. (2014) possui dois níveis de paralelismo: processos e *threads*. O processo mestre gera o *template* da impressão digital de prova (T_p) e coleta os resultados parciais gerados pelos processos escravos. Seja $T = \{t_1, t_2, ..., t_n\}$ o conjunto de *templates* de impressões digitais armazenados na base de dados. Seja p o número de nós do *cluster* e h o número de *threads* por nó. Cada escravo E_i, onde $i > 0$ e $i \leq p$, carrega em memória uma partição $P_{E_i} \subseteq T$ da base de dados, tal que $|P_{E_i}| = n/p$, e realiza o confronto de T_p contra P_{E_i}. Seguindo para o próximo nível de paralelismo, tem-se que cada E_i é formado por

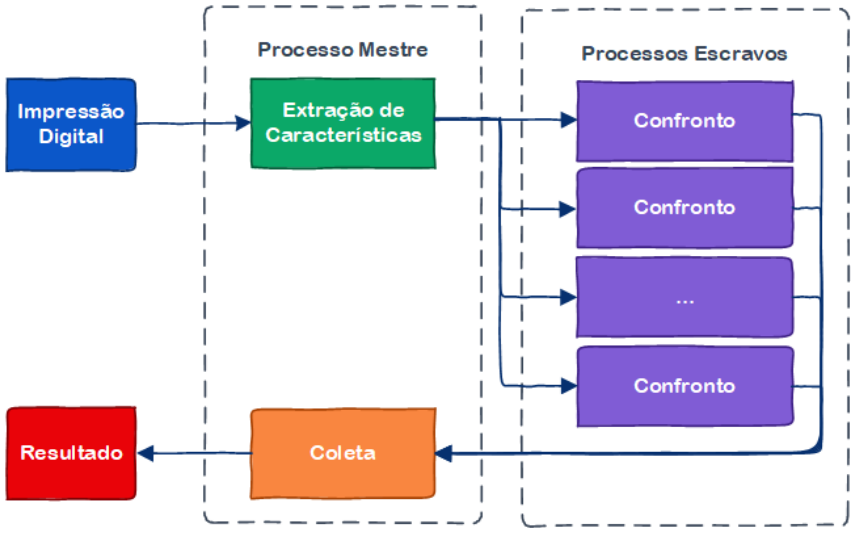

Figura 11 – Modelo de processamento distribuído proposto por Peralta et al. (2014).
Fonte: Adaptado de Peralta et al. (2014)

uma ou mais *threads*, $E_i = \{\tau_1, \tau_2, ..., \tau_h\}$, fazendo com que P_{E_i} seja particionado para que cada *thread* $\tau_j \in E_i$, realize o confronto de T_p contra uma partição $P_{\tau_j} \subseteq P_{E_i}$, tal que $|P_{\tau_j}| = n/(ph)$.

O esquema proposto garante que o confronto seja executado em dois níveis de paralelismo, acelerando a execução. Além disso, permite que o sistema seja dimensionado para qualquer tipo de *hardware*, ambiente e tamanho de banco de dados, apenas ajustando p e h, obtendo um ganho máximo de performance. Nos experimentos foi utilizada uma base de dados contendo 400.000 impressões digitais. Como não existem bases públicas de impressões digitais desta magnitude, os autores utilizaram o SFinGe (MALTONI, 2004) para gerar impressões digitais sintéticas e incluíram duas bases públicas: NIST *Special Database* 4 (WATSON, 1992), que contém 2.000 pares de impressões digitais roladas; e NIST *Special Database* 14 (WATSON, 1993), que contém 27.000 pares de impressões digitais roladas. Foi utilizado um *cluster* híbrido (Figura 12) contendo até 12 nós, onde cada nó possuía as seguintes configurações:

- Processadores: 2 × Intel Xeon E5-2620 2.00 GHz
- Núcleos: 6 por processador (12 *threads*)
- Rede: Gigabit Ethernet (1 Gbps)
- Memória RAM: 64 GB

Peralta et al. (2014) concluem que, utilizando o *framework* proposto é possível manter o tempo de identificação constante não obstante o crescimento da base de dados. Para tanto, basta adicionar mais *hardware* na mesma proporção deste crescimento. Os experimentos mostraram também que esta abordagem obteve bons resultados, independentemente do algoritmo de confronto e do tipo de *template* utilizado.

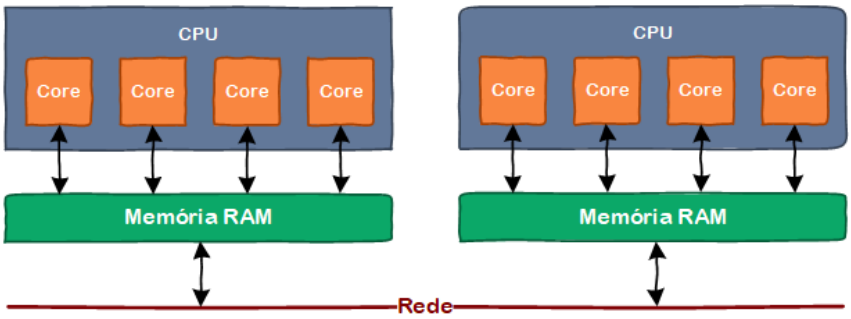

Figura 12 – Arquitetura de *cluster* híbrido utilizada por Peralta et al. (2014). Em um *cluster* híbrido, uma tarefa é dividida em diversos processos e cada processo é executado em um nó diferente (primeiro nível de paralelismo). Estes processos se comunicam através de MPI. Novamente, cada processo pode ser dividido em diversas *threads* de execução que podem se comunicar através de memória compartilhada, que é mais rápida que MPI. A performance máxima é atingida geralmente quando cada nó de computação executa um único processo que contém uma ou duas *threads* por núcleo do nó (PERALTA et al., 2014).
Fonte: Adaptado de Peralta et al. (2014)

Peralta et al. (2016) trazem em seu trabalho uma extensão para a abordagem descrita em Peralta et al. (2014), utilizando uma fusão entre dois algoritmos de confronto e duas impressões digitais. Esta nova proposta tem como objetivo conseguir um equilíbrio entre acurácia, eficiência e escalabilidade nas operações com grandes bases de dados.

A Figura 13 apresenta o plano de processos proposto. Seja $T = \{t_1, t_2, ..., t_n\}$ o conjunto de *templates* de impressões digitais armazenados na base de dados, T_p o *template* de prova e $s_{fast}(T_p, t_i)$ o escore de similaridade entre o *template* de prova e um *template* $t_i \in T$ gerado pelo algoritmo de confronto rápido. Na primeira fase, um algoritmo de confronto rápido (JIANG; YAU, 2000) é aplicado em T para criar um conjunto de candidatos C.

Dois critérios podem ser utilizados para gerar este conjunto (PERALTA et al., 2016):

- **Classificação**: são selecionados os r candidatos com maiores escores de similaridade de modo que $|C| = r$.
- **Limiar**: todos os *templates* $t_i \in T$ que obtiverem escore de similaridade $s_{fast}(T_p, t_i)$ maior ou igual a um limiar fixo ϕ são adicionados a C, logo $C = \{t_i \mid s_{fast}(T_p, t_i) \geq \phi\}$.

Seja $s_{accurate}(T_p, t_j)$ o escore de similaridade entre o *template* de prova e um *template* $t_j \in C$ gerado pelo algoritmo de confronto preciso. Na próxima fase, um algoritmo preciso de confronto (CAPPELLI; FERRARA; MALTONI, 2010) é aplicado no conjunto C e o candidato com maior $s_{accurate}(T_p, t_j)$ é retornado (PERALTA et al., 2016).

Para aumentar a acurácia do procedimento, Peralta et al. (2016) propõem utilizar duas impressões digitais no confronto, realizando posteriormente a fusão de escores pela média desses valores. Os experimentos foram conduzidos utilizando o mesmo ambiente de Peralta et al. (2014). Entretanto, a maior base utilizada foi gerada pelo SFinGe e continha 50.000 pares de impressões

digitais. O tempo de resposta para uma identificação contra a base sintética utilizando os 12 nós do *cluster* foi de apenas 0.44 segundos.

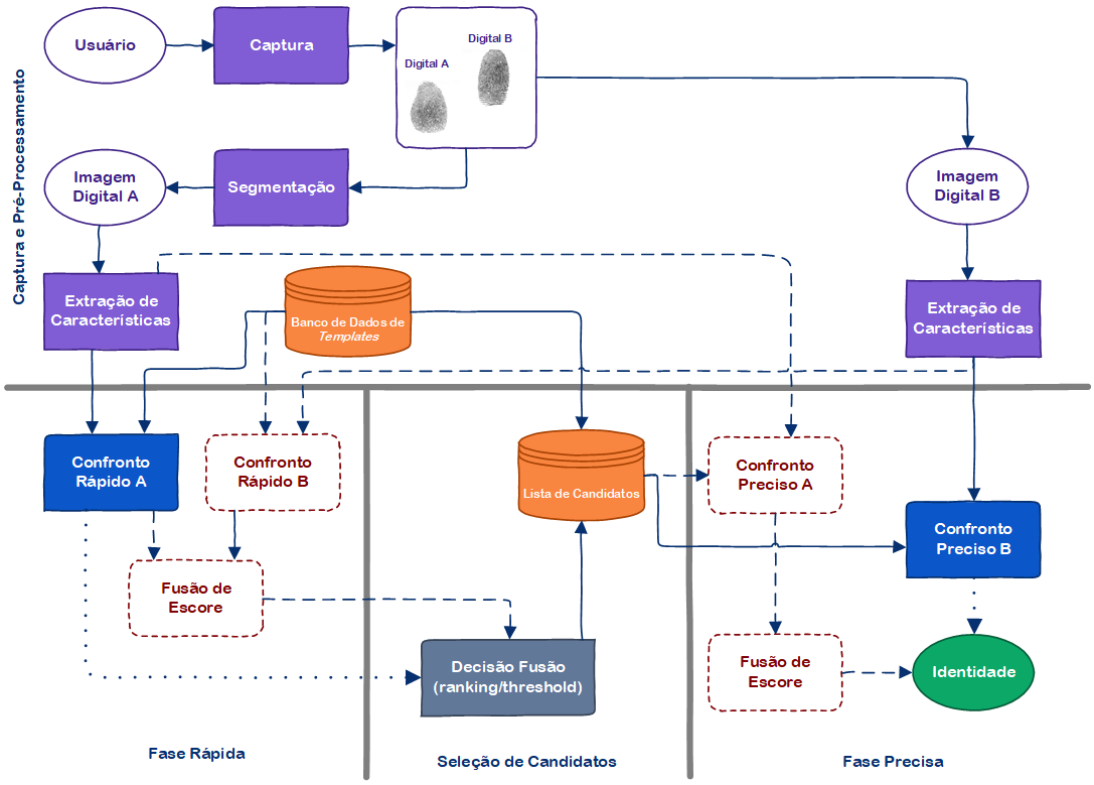

Figura 13 – Plano de processos da abordagem apresentada por Peralta et al. (2016). As linhas tracejadas correspondem ao fluxo utilizando a variante de dois dedos. Nesta variante, as digitais A e B são utilizadas no confronto rápido e no confronto preciso. As linhas pontilhadas correspondem ao fluxo utilizando a variante de um dedo. Nesta variante, apenas a digital A é utilizada no confronto rápido e somente a digital B é utilizada no confronto preciso. As linhas contínuas são caminhos que são sempre seguidos.
Fonte: Adaptado de Peralta et al. (2016)

Peralta et al. (2017) descrevem uma metologia (Figura 14) para adaptar algoritmos de confronto de impressões digitais baseados em minúcias para os *frameworks* Big Data Apache Hadoop (The Apache Software Foundation, 2017a) e Apache Spark (The Apache Software Foundation, 2017d). Um algoritmo de confronto clássico geralmente possui duas fases: local e global (consolidação) (JIANG; YAU, 2000; CAPPELLI; FERRARA; MALTONI, 2010). Sejam L_i^P e L_j^T os conjuntos de estruturas locais gerados a partir do *template* de prova e do *template* de referência e q_{ij} o escore de similaridade entre esses conjuntos. Os autores introduzem o conceito de escore parcial s, que representa a similaridade entre os dois sub-conjuntos não vazios $L_{ik}^P \subseteq L_i^P$ e $L_{jk}^T \subseteq L_j^T$, na forma $s(L_{ik}^P, L_{jk}^T)$. Estes escores parciais são calculados independentemente e combinados posteriormente. Os experimentos, mostraram que a abordagem proposta consegue taxas de processamento maiores do que aquelas apresentadas por Peralta et al. (2014).

Peralta et al. (2014), Peralta et al. (2016) e Peralta et al. (2017) utilizaram em seus experimentos impressões digitais sintéticas geradas pelo SFinGe. É preciso salientar que Gottschlich e Huckemann (2014) afirmam em seu trabalho que, qualquer resultado obtido utilizando impres-

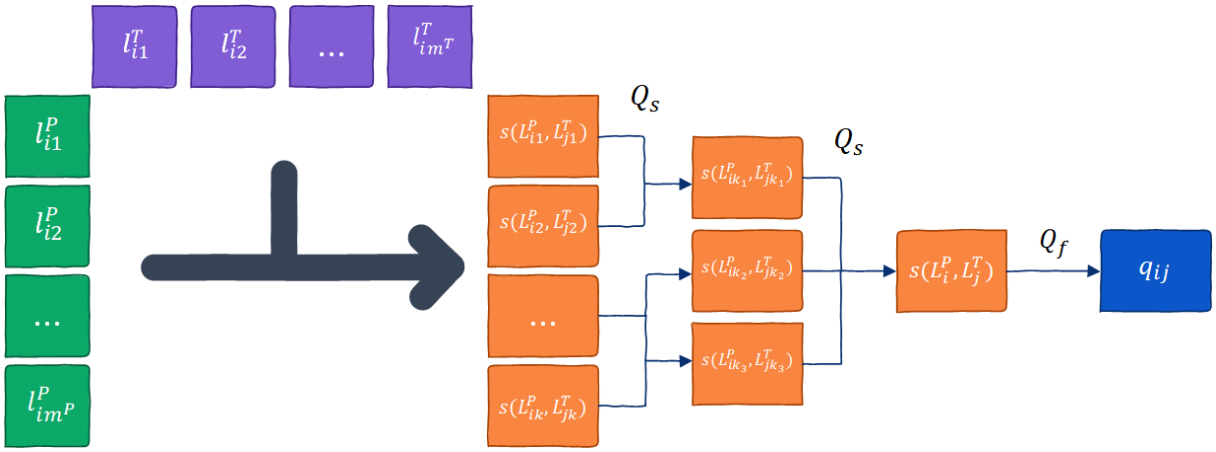

Figura 14 – Metodologia de decomposição proposta por Peralta et al. (2017). O processo começa com os conjuntos de estruturas locais $L^P_i = \{l^P_{i1}, l^P_{i2}, ..., l^P_{imP}\}$ e $L^T_j = \{l^T_{j1}, l^T_{j2}, ..., l^T_{jmT}\}$. Essas estruturas são agrupadas arbitrariamente em conjuntos da forma $L^P_{ik} \subseteq L^P_i$ e $L^T_{jk} \subseteq L^T_j$. Estes conjuntos são utilizados no cálculo do escore de similaridade $s(L^P_{ik}, L^T_{jk})$. Os escores parciais são agregados através da aplicação sucessiva de uma função de Q_s até que se obtenha um único escore parcial, que representa a similaridade entre os conjuntos L^P_i e L^T_j. Por fim, uma outra função Q_f é aplicada para que se obtenha o valor final q_{ij} (PERALTA et al., 2017).
Fonte: Adaptado de Peralta et al. (2017)

sões digitais geradas pelo SFinGe podem não refletir a performance do sistema em um cenário real, pois existem diferenças fundamentais entre o processo de formação de impressões digitais reais e aquelas sintetizadas por este *software*. Durante o processo de geração das impressões digitais sintéticas pelo SFinGe (Figura 15), é possível configurar o número de sementes utilizadas, de forma que, quanto menor o número de sementes, menos minúcias são geradas (MALTONI et al., 2009). O tempo de resposta de um algoritmo de confronto de impressões digitais depende do número de minúcias dos *templates*. Sendo assim, é possível manipular os tempos de resposta do sistema através do uso de bases sintéticas com um menor ou maior número de minúcias.

3.2 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo descreveu um levantamento dos trabalhos que apresentam abordagens de implementação de um sistema biométrico capaz de operar com grandes bases de dados. A solução proposta se fundamenta na generalização dos conceitos obtidos destes trabalhos, principalmente os de Peralta et al. (2014) e Peralta et al. (2016), para implementar uma central de confrontos escalável, altamente paralelizável e capaz de operar com grandes bases de dados. O modelo de processamento distribuído adotado neste trabalho é muito similar ao de Peralta et al. (2014), onde tem-se um um controlador e um conjunto de nós de processamento. Na solução apresentada neste trabalho foram utilizados dois algoritmos de confronto, um rápido e outro preciso, como em Peralta et al. (2016). A grande diferença é a utilização de um *framework* de computação em memória para implementar o sistema proposto e o uso de uma base de dados real para validar tal sistema.

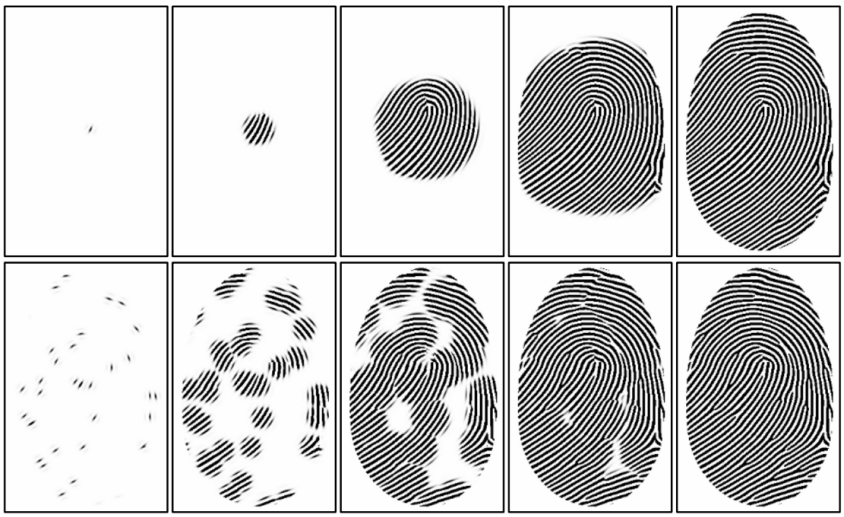

Figura 15 – Processo de geração de impressões digitais sintéticas pelo SFinGe. Na linha superior, a impressão digital foi gerada a partir de uma única semente e contém poucas minúcias. Na linha inferior, a impressão digital foi gerada a partir de várias sementes e contém um maior número de minúcias.

Fonte: Maltoni et al. (2009)

4 METODOLOGIA

A central de confrontos é um dos módulos mais complexos em um sistema biométrico de larga escala, principalmente quando são realizadas operações de identificação. Visando minimizar esta complexidade, a seguir é formalizada a abordagem de implementação proposta.

4.1 CENTRAL DE CONFRONTOS DISTRIBUÍDA E ESCALÁVEL

Esta solução tem como objetivo implementar uma central de confrontos para um ABIS que possa atender aos casos de uso mais comuns: identificação civil, criminal e forense. A especificação descrita a seguir está inserida neste escopo de aplicações, porém algumas simplificações são feitas para facilitar o formalismo. Na Seção 4.2 estas simplificações são removidas e a solução final de implementação é apresentada.

Seja $ConjRegsCadastrados = \{R_1, R_2, ..., R_n\}$ o conjunto dos n registros biométricos cadastrados no sistema. É possível particionar este conjunto de acordo com o tipo do registro $R_i \in ConjRegsCadastrados$. Seja $ConjRegsCivis \subseteq ConjRegsCadastrados$ o conjunto de registros civis e $ConjRegsCriminais \subseteq ConjRegsCadastrados$ o conjunto de registros criminais. A divisão lógica dos registros faz com que as pesquisas nos registros civis, que tendem a ser mais demoradas pois $|ConjRegsCivis| \gg |ConjRegsCriminais|$, sejam independentes das pesquisas nos registros criminais. Esta independência faz com que a taxa de processamento do sistema seja maior, visto que as pesquisas criminais não precisam aguardar o término de uma pesquisa civil para serem processadas.

Cada registro biométrico $R_i \in ConjRegsCadastrados$ é definido por $R_i = (K_i, V_i)$, um par chave e valor. A chave do registro K_i é definida por $K_i = (\iota, \varsigma)$, onde ι é o identificador único do indivíduo e ς é o número sequencial do registro. Este número sequencial é importante, pois através dele é possível manter um histórico das capturas biométricas de um mesmo indivíduo. O valor do registro V_i é definido por $V_i = (\sigma, \alpha, T)$, onde σ é o sexo do indivíduo, α é o ano de nascimento do indivíduo e $T = \{t_1, t_2, ...t_r\}$ é o conjunto de *templates*. O sexo e o ano de nascimento do indivíduo são utilizados como critérios de redução do espaço de busca no conjunto de registros.

Supondo que a central de confrontos é composta por η nós, onde cada nó é capaz de processar τ *threads* simultaneamente. É possível criar grupos de processamento, de forma que as pesquisas em $ConjRegsCivis$ não concorram com aquelas em $ConjRegsCriminais$. Sendo assim, tem-se $ConjNósCivis = \{nó_1, nó_2, ..., nó_{\eta_{civ}}\}$ o conjunto de nós dedicados ao processamento de pesquisas em $ConjRegsCivis$ e $ConjNósCriminais = \{nó_{\eta_{civ}+1}, nó_{\eta_{civ}+2}, ..., nó_{\eta}\}$ o conjunto de nós dedicados ao processamento de pesquisas em $ConjRegsCriminais$, de modo que $\eta_{cri} = |ConjNósCriminais|$ e $\eta_{civ} = |ConjNósCivis|$.

Algoritmo 1 – Operação *map* de uma identificação em $ConjRegsCivis$

Entrada: Conjunto de *Templates* de Prova T_p, Prioridade ρ
Saída: Conjunto de Subtarefas J
1: $J \leftarrow \emptyset$;
2: **para** $k = 1$ **até** p_{civ} **faça**
3: $\quad j_k \leftarrow CriaSubTarefa(T_p, k, \rho)$;
4: $\quad J \leftarrow J \cup j_k$;
5: **fim para**
6: **retorna** J;

Fonte: Autoria própria

Para tornar o processamento mais efetivo, é necessário dividir os conjuntos de registros em partições menores e distribuí-las entre os respectivos nós dedicados. Seja $PCivil \subseteq ConjRegsCivis$ uma partição dos registros civis e $PCriminal \subseteq ConjRegsCriminais$ uma partição dos registros criminais. São criadas p_{civ} partições $PCivil$ para $ConjRegsCivis$ e p_{cri} partições $PCriminal$ para $ConjRegsCriminais$ tal que $p_{civ} \geq \tau\eta_{civ}$ e $p_{cri} \geq \tau\eta_{cri}$, de forma a se obter máxima utilização dos recursos computacionais. Logo, $ConjRegsCivis = PCivil_1 \cup PCivil_2 \cup ... \cup PCivil_{p_{civ}}$ e $ConjRegsCriminais = PCriminal_1 \cup PCriminal_2 \cup ... \cup PCriminal_{p_{cri}}$.

É preciso garantir que as cardinalidades das partições sejam aproximadamente iguais, $|PCivil_1| \approx |PCivil_2| \approx ... \approx |PCivil_{p_{civ}}|$ e $|PCriminal_1| \approx |PCriminal_2| \approx ... \approx |PCriminal_{p_{cri}}|$, e o número de partições em cada nó também seja aproximadamente igual para que a carga de processamento seja distribuída uniformemente. Desta forma, a capacidade de armazenamento e o poder de processamento aumenta linearmente com a adição de nós ao *cluster*.

As funcionalidades da solução proposta são: identificação, verificação, inclusão e exclusão de registros.

4.1.1 Identificação

Seja $I_{civ}(T_p, \rho)$ uma tarefa de identificação em $ConjRegsCivis$. Esta tarefa recebe como entrada um conjunto de *templates* de prova T_p e a prioridade da tarefa $\{\rho \in \mathbb{Z} \mid 0 \leq \rho \leq 10\}$. Efetua uma operação *map* (Algoritmo 1) onde é criado um conjunto de subtarefas $J = \{j_1, j_2, ..., j_{p_{civ}}\}$. Cada subtarefa $j_k \in J$ realiza o confronto de T_p contra uma partição $PCivil_k \subseteq ConjRegsCivis$, sendo executada pelo nó que contém tal partição e, retorna um conjunto ordenado de resultados com maiores escores de similaridade $M_{j_k} = \{m_1, m_2, ..., m_u\}$, com $|M_{j_k}| \leq NumMaxCandidatos$ e $m_j \in M_{j_k}$ é definido por $m_j = (K_i, \varepsilon_{pi})$, onde K_i é a chave do registro de referência $R_i \in PCivil_k$ e ε_{pi} é o escore de similaridade entre T_p e R_i (Algoritmo 2).

Quando todas as subtarefas J terminam, a operação *reduce* (Algoritmo 3) é executada. Nela, o conjunto de resultados de todas as subtarefas $M_J = M_{j_1} \cup M_{j_2} \cup ... \cup M_{j_{p_{civ}}}$, é ordenado pelo escore de similaridade em ordem decrescente e avaliado para produzir o conjunto final de resultados M, com $|M| \leq NumMaxCandidatos$.

Algoritmo 2 – Subtarefa j_k em $ConjRegsCivis$

Entrada: Conjunto de Templates de Prova T_p, Partição para Confronto $PCivil_k$
Saída: Conjunto de Resultados M_{j_k}

1: $NumMaxCandidatos \leftarrow 100$;
2: $M_{j_k} \leftarrow \emptyset$;
3: **para cada** $R_i \in PCivil_k$ **faça**
4: $\quad m_i \leftarrow RealizaConfronto(T_p, R_i)$;
5: $\quad M_{j_k} \leftarrow M_{j_k} \cup m_i$;
6: **fim para**
7: $M_{j_k} \leftarrow OrdenaResultadosEscore(M_{j_k})$;
8: **se** $|M_{j_k}| > NumMaxCandidatos$ **então**
9: $\quad M_{max} \leftarrow \emptyset$;
10: $\quad$ **para** $n = 1$ **até** $NumMaxCandidatos$ **faça**
11: $\quad\quad M_{max} \leftarrow M_{max} \cup m_n \in M_{j_k}$;
12: $\quad$ **fim para**
13: $\quad M_{j_k} \leftarrow M_{max}$;
14: **fim se**
15: **retorna** M_{j_k};

Fonte: Autoria própria

4.1.2 Verificação

Seja $V_{civ}(T_p, K_v)$ uma tarefa de verificação (Algoritmo 4) em $ConjRegsCivis$ e μ o limiar utilizado para tomada de decisão. Esta tarefa recebe como entrada um conjunto de *templates* de prova T_p e a chave K_v do registro de referência R_v a ser verificado. O registro R_v é recuperado através de sua chave K_v e o confronto de T_p e R_v é realizado. O resultado da operação de verificação é ω_v, um booleano que indica se a verificação foi positiva (verdadeiro) ou negativa (falso). Esta tomada de decisão é baseada no escore de similaridade ε_{pv} entre T_p e R_v, de forma que:

$$\omega_v = \begin{cases} verdadeiro & \text{se } \varepsilon_{pv} \geq \mu \\ falso & \text{se } \varepsilon_{pv} < \mu \end{cases} \tag{1}$$

4.1.3 Inclusão e Remoção de Registros

Seja $A_{civ}(R_i)$ uma tarefa de inclusão de registro em $ConjRegsCivis$. Esta tarefa recebe como entrada o registro $R_i = (K_i, V_i)$ a ser adicionado. A partição $PCivil_k \subseteq ConjRegsCivis$ em que o registro R_i deve ser adicionado é calculada. Tal registro é então enviado para o nó que contém tal partição, de forma que $PCivil_k \leftarrow PCivil_k \cup R_i$.

Seja $D_{civ}(K_i)$ uma tarefa de exclusão de registro em $ConjRegsCivis$. Esta tarefa recebe como entrada a chave K_i do registro a ser excluído. A partição $PCivil_k \subseteq ConjRegsCivis$ em que

Algoritmo 3 – Operação *reduce* de uma identificação em $ConjRegsCivis$

Entrada: Conjunto de Resultados de Todas as Subtarefas M_J
Saída: Conjunto Final de Resultados M

$\lambda \leftarrow 0.6$ // Constante utilizada para filtrar lista de candidatos.
$M \leftarrow \emptyset$;
para cada $m_i \in M_J$ **faça**
    $M \leftarrow M \cup m_i$;
fim para
$M \leftarrow OrdenaResultadosEscore(M)$;
se $|M| > k$ **então**
    $M_k \leftarrow \emptyset$;
    para $i = 1$ **até** k **faça**
        $M_k \leftarrow M_k \cup m_i \in M$;
    fim para
    $M \leftarrow M_k$;
fim se
se $|M| > 0$ **então**
    $M_{filt} \leftarrow \emptyset$;
    $\varepsilon_{max} \leftarrow GetEscore(m_1 \in M)$;
    $M_{filt} \leftarrow M_{filt} \cup m_1$;
    para $i = 2$ **até** $|M|$ **faça**
        se $GetEscore(m_i \in M) \geq \lambda * \varepsilon_{max}$ **então**
            $M_{filt} \leftarrow M_{filt} \cup m_i \in M$;
        fim se
    fim para
    $M \leftarrow M_{filt}$;
fim se
retorna M;

Fonte: Autoria própria

Algoritmo 4 – Tarefa de verificação V_{civ} **em** $ConjRegsCivis$

Entrada: Conjunto de *Templates* de Prova T_p, Chave do registro a ser verificado K_v
Saída: Resultado da verificação M_v

$R_v \leftarrow RecuperaRegistro(K_v)$;
$m \leftarrow RealizaConfronto(T_p, R_v)$
se $GetEscore(m) \geq \mu$ **então**
    $\omega_v \leftarrow$ **verdadeiro** // genuíno
senão,
    $\omega_v \leftarrow$ **falso** // impostor
fim se
retorna ω_v;

Fonte: Autoria própria

o registro R_i está armazenado, é calculada utilizando a chave K_i. Envia a solicitação de remoção para o nó que contém tal partição, de forma que $PCivil_k \leftarrow PCivil_k - \{R_i\}$.

4.2 IMPLEMENTAÇÃO

Nesta seção será apresentada a abordagem utilizada para implementar a arquitetura (Figura 16) apresentada na Seção 4.1 utilizando o Apache Ignite.

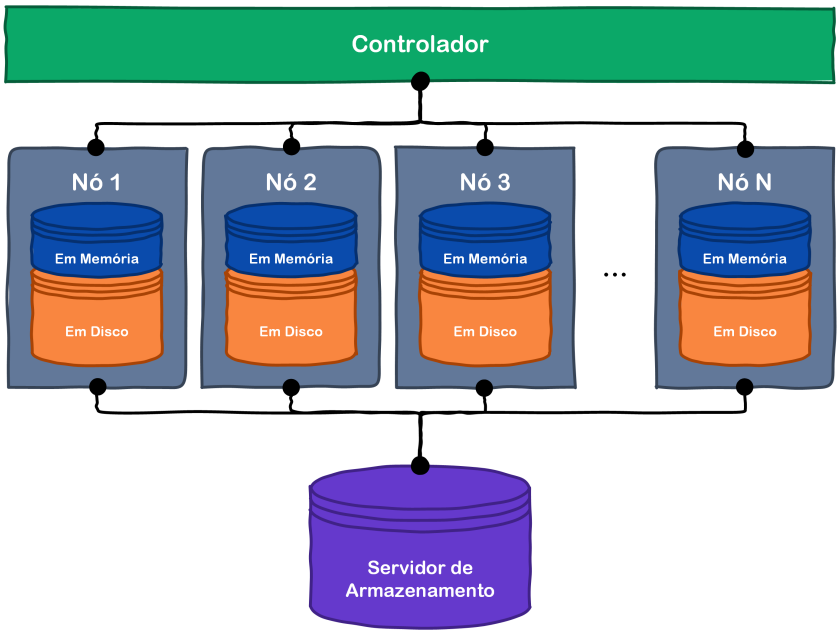

Figura 16 – Arquitetura da central de confrontos proposta. Formada por um conjunto de nós que fazem cache em memória e em disco (memória durável) dos *templates* do servidor de armazenamento e um controlador que recebe requisições, despacha o processamento no *cluster*, processa os resultados e envia respostas.

Fonte: Autoria própria

Pesquisas de identificação forense são mais demoradas, pois o algoritmo de confronto utilizado deve ser mais robusto devido à baixa qualidade e grandes distorções presentes nas amostras apresentadas ao sistema (MATHEW; THOMAS; KIZHAKKETHOTTAM, 2015; SANKARAN; VATSA; SINGH, 2014). Sendo assim, faz sentido criar um grupo de nós exclusivo para processamento de tais pesquisas.

Foram criados quatro grupos de nós no *cluster*: civil, civil forense, criminal e criminal forense. Em cada grupo foi instanciado um *cache* particionado. Neste tipo de *cache*, o Apache Ignite divide igualmente o conjunto de dados em partições, e estas são distribuídas entre todos os nós participantes do grupo, utilizando o algoritmo *Rendezvous* (*Highest Random Weight*) (THALER; RAVISHANKAR, 1998). O uso deste algoritmo faz com que a distribuição das partições entre os nós não seja exatamente igual. Entretanto, ele garante que durante mudanças de topologia do *cluster*, não ocorra migração de partições entre os nós existentes. Existe movimentação de dados somente para o nó adicionado ou a partir do nó que deixou a topologia (The Apache Software Foundation, 2017c). Isto faz com que o tráfego de dados na rede seja

reduzido. Nos grupos civil e civil forense os *caches* contém *ConjRegsCivis* e nos grupos criminal e criminal forense os *caches* contém *ConjRegsCriminais*.

Os registros estão armazenados em um banco de dados centralizado. Durante a primeira inicialização do *cluster*, estes dados são carregados para a memória e para o disco dos nós participantes. Este processo de carga é implementado utilizando a API de armazenamento persistente do Apache Ignite. Foi criada a classe *BiometricCacheStore* que estende a classe *CacheStoreAdapter* desta API. Nela são implementados os métodos que recuperam, escrevem e apagam registros no banco de dados e o método que carrega o *cache* inteiro. Este último é executado em todos os nós da topologia durante o processo de inicialização. O uso desta API permite que registros sejam lidos do banco de dados quando não estiverem disponíveis no *cache* em memória ou em disco e, da mesma forma, sejam automaticamente persistidos (adicionados, atualizados ou removidos) no banco de dados quando forem alterados no *cache*. Sendo assim, as operações de inclusão e remoção de registros são implementadas através dos métodos *put* e *remove*, já disponíveis no módulo de *cache* distribuído do Apache Ignite.

O processo de identificação (Figura 17) é implementado utilizando a API de computação distribuída do Apache Ignite, que é uma abstração simplificada de um *MapReduce* (DEAN; GHEMAWAT, 2008) em memória. Foram criadas duas classes:

- ***IdentificationTask***: Implementa a interface *ComputeTask*, onde são definidos os métodos *map* e *reduce*. No método *map* (Algoritmo 1) é criado o conjunto de subtarefas, definido o mapeamento destas para os nós que as executarão e configurada a prioridade de execução da tarefa. No método *reduce* (Algoritmo 3) os resultados de todas as subtarefas são agregados, ordenados e filtrados. São atributos desta classe o conjunto de *templates* de prova, a prioridade e um conjunto de parâmetros de busca que contém a base-alvo da pesquisa (civil ou criminal), o tipo da pesquisa (comum ou forense), número máximo de candidatos retornados e critérios de redução do espaço de busca (idade, sexo, identificador biométrico etc.).

- ***MatchJob***: Estende a classe *ComputeJobAdapter*, onde é definido o método *execute*. Neste método é implementado o confronto biométrico (Algoritmo 2). Para pesquisas forenses, é utilizado apenas o algoritmo de confronto preciso (Algoritmo 5). Já para as pesquisas comuns, um algoritmo rápido é aplicado para gerar uma lista de possíveis candidatos e posteriormente o algoritmo preciso é aplicado somente neste conjunto (Algoritmo 6), processo este que foi inspirado em Peralta et al. (2016). São atributos desta classe o conjunto de *templates* de prova, o conjunto de parâmetros de busca e a partição em que será realizado o confronto. Os *MatchJobs* são criados pela *IdentificationTask* no controlador e enviados para os nós de execução. Quando chegam nos respectivos nós, são ordenados por prioridade antes de serem submetidos ao *pool* de *threads* de execução. Isto é controlado pela classe *PriorityQueueCollisionSpi* do Apache Ignite.

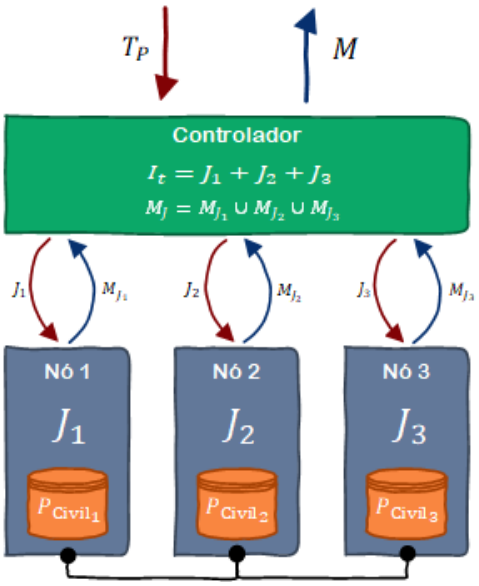

Figura 17 – Esquemático do processo de identificação em $ConjRegsCivis = PCivil_1 \cup PCivil_2 \cup PCivil_3$. O controlador recebe um conjunto de *templates* de prova T_p, cria uma subtarefa para cada $PCivil_k$ e as envia para execução no nó que contenha tal partição. Os resultados de cada subtarefa são encaminhados para o controlador que os processa e monta o conjunto final de resultados M.

Algoritmo 5 – Operação *RealizarConfronto* para identificação forense

Entrada: Conjunto de *Templates* de Prova T_p, Registro de Referência R_i
Saída: Resultado da identificação m_i
1: $m_r \leftarrow RealizaConfrontoPreciso(T_p, R_i)$
2: **retorna** m_r;

Fonte: Autoria própria

Algoritmo 6 – Operação *RealizarConfronto* para identificação comum

Entrada: Conjunto de *Templates* de Prova T_p, Registro de Referência R_i
Saída: Resultado da identificação m_i
1: $m_r \leftarrow RealizaConfrontoRápido(T_p, R_i)$
2: **se** $GetEscore(m_r) \geq \mu_r$ **então**
3: $\quad m_r \leftarrow RealizaConfrontoPreciso(T_p, R_i)$
4: **senão,**
5: $\quad m_r \leftarrow null$
6: **fim se**
7: **retorna** m_r;

Fonte: Autoria própria

A operação de verificação é implementada utilizando uma funcionalidade do Apache Ignite que permite a execução de uma tarefa no nó em que o registro de referência envolvido na operação está armazenado. Isto reduz os tempos de serialização de dados e o tráfego de rede no *cluster*. Foi criada a classe *VerificationCallable* que implementa a interface *IgniteCallable*. O método *call* recupera o registro de referência, realiza o confronto e toma uma decisão baseada no escore de similaridade (Algoritmo 4).

4.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo formalizou a arquitetura proposta da central de confrontos de um ABIS capaz de operar em grandes bases de dados. A solução está baseada nos casos de uso mais comuns: identificação civil, criminal e forense. Além disso, apresentou detalhes da implementação desta arquitetura utilizando a API do Apache Ignite para carregar os registros de um servidor de armazenamento centralizado e realizar as operações de inclusão e remoção de registros, identificação e verificação.

5 AVALIAÇÃO EXPERIMENTAL

Neste capítulo a abordagem proposta é avaliada. O ambiente em que os experimentos foram conduzidos é apresentado, bem como os resultados obtidos. O objetivo destes experimentos é validar a escalabilidade do sistema. Ao final, uma análise destes resultados e algumas considerações são descritas.

5.1 CONFIGURAÇÃO DOS EXPERIMENTOS

Esta seção descreve o ambiente e a base de dados utilizados neste trabalho.

5.1.1 Ambiente de *hardware* e *software*

O *cluster* utilizado nos experimentos é composto por servidores *Dell PowerEdge* R430, contendo até 16 nós, com as seguintes configurações:

- 1 servidor tipo 1:
  - Processadores: 2 × Intel Xeon E5-2620 v3 2.40 GHz
  - Núcleos: 12 (24 *threads*)
  - Rede: Gigabit Ethernet (2 Gbps)
  - Memória RAM: 16 GB
  - Sistema Operacional: Windows Server 2012 R2
- 11 servidores tipo 2:
  - Processadores: 2 × Intel Xeon E5-2650 v3 2.30 GHz
  - Núcleos: 20 (40 *threads*)
  - Rede: Gigabit Ethernet (2 Gbps)
  - Memória RAM: 64 GB
  - Sistema Operacional: Windows Server 2012 R2
- 4 servidores tipo 3:
  - Processadores: 2 × Intel Xeon E5-2470 v2 2.40 GHz
  - Núcleos: 20 (40 *threads*)
  - Rede: Gigabit Ethernet (2 Gbps)
  - Memória RAM: 48 GB

- Sistema Operacional: Windows Server 2012 R2

Em cada nó são armazenadas 80 partições do conjunto de registros, que são processadas por 80 *threads*. O tamanho deste *pool* de Este número de *threads* é a configuração padrão do Apache Ignite que considera duas vezes o número de processadores lógicos. Desta forma, em uma topologia contendo 3 nós, existirão 240 partições e 240 *threads* para processamento.

A central de confrontos proposta é implementada utilizando Java e o Apache Ignite 2.2.0. Os algoritmos de confronto de impressões digitais utilizados nos experimentos são de propriedade da Antheus Tecnologia Ltda. (2017) e são implementados em C++.

O nó controlador é executado no servidor tipo 1. As requisições são recebidas e os resultados são enviados através de mensagens AMQP. Nesta solução foi utilizado o RabbitMQ (Pivotal, 2017) como *message broker*. Todos os experimentos conduzidos neste trabalho foram realizados em um ambiente real de operação, conforme Figura 18. As solicitações dos clientes são recebidas pelo respectivo *web service* e enviadas para serem processadas pelo respectivo *workflow*. Durante seu fluxo de operação, o *workflow* pode enviar requisições para a central de confrontos. Estas requisições são processadas e os resultados são encaminhados para o plano de processos, que continua sua execução e responde ao *web service*. Os resultados podem ser então consultados pelos clientes.

Todos os experimentos foram executados uma vez em um *cluster* dedicado.

5.1.2 Base de Dados

Para avaliar a escalabilidade do sistema, é necessário uma base de dados biométricos grande. Porém, não existe base pública contendo impressões digitais (registros decadatilares) com tamanho suficiente para realizar os experimentos. Foi utilizada uma base de dados privada contendo 5.313.097 registros biométricos. Destes, 5.287.958 são registros civis e 25.139 são registros criminais.

Os registros civis contém apenas impressões digitais e os registros criminais contém impressões digitais, face e palmar. Apesar da central de confrontos implementada suportar íris, esta não foi utilizada nos experimentos pois a base de dados não contém registros com este tipo de biometria.

Os *templates* de impressões digitais foram gerados utilizando o Agora SDK (Antheus Tecnologia Ltda., 2017). A posição do dedo também é utilizada como critério de redução do espaço de busca.

5.2 EXPERIMENTOS

Esta seção descreve os experimentos realizados e apresenta os resultados obtidos.

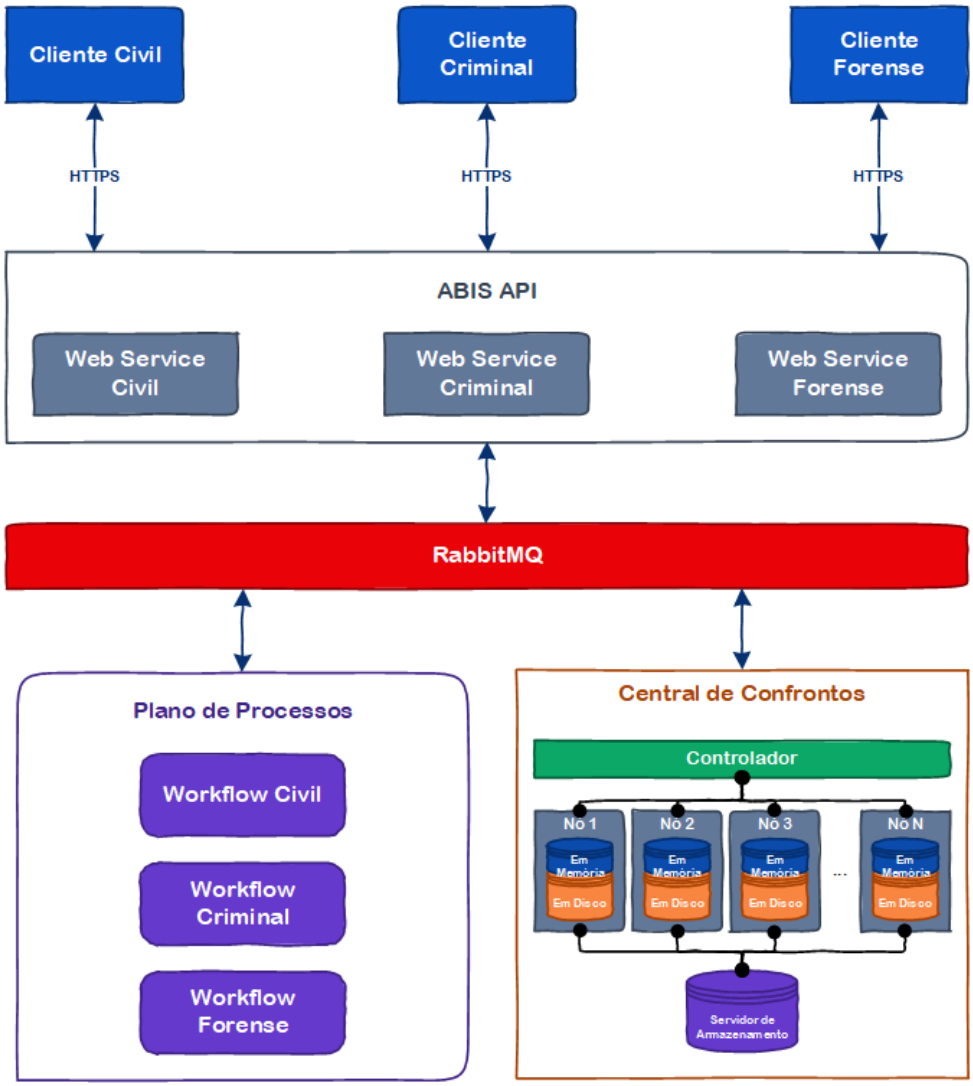

Figura 18 – Arquitetura do ABIS utilizado nos experimentos.

5.2.1 Experimento I

No primeiro experimento foram alocados até 15 nós para o *cluster* de processamento e um nó para o controlador. Para que todos os *templates* do conjunto de registros civis pudessem ser carregados em memória, a topologia mínima adotada foi de 3 nós. Foram avaliadas 5 topologias:

- 3 nós (3 servidores do tipo 2).
- 6 nós (6 servidores do tipo 2).
- 9 nós (9 servidores do tipo 2).
- 12 nós (11 servidores do tipo 2 e 1 servidor do tipo 3).
- 15 nós (11 servidores do tipo 2 e 4 servidores do tipo 3).

Um conjunto contendo 10 impressões digitais (Figura 19) de um mesmo indivíduo foi utilizado como amostra de prova. Foram enviadas 11 identificações para cada topologia, sendo

que a primeira continha apenas o polegar direito, a segunda o indicador direito, a terceira o médio direito e assim sucessivamente, até que a décima identificação continha o mínimo esquerdo. A décima primeira identificação continha o conjunto completo (10 impressões digitais).

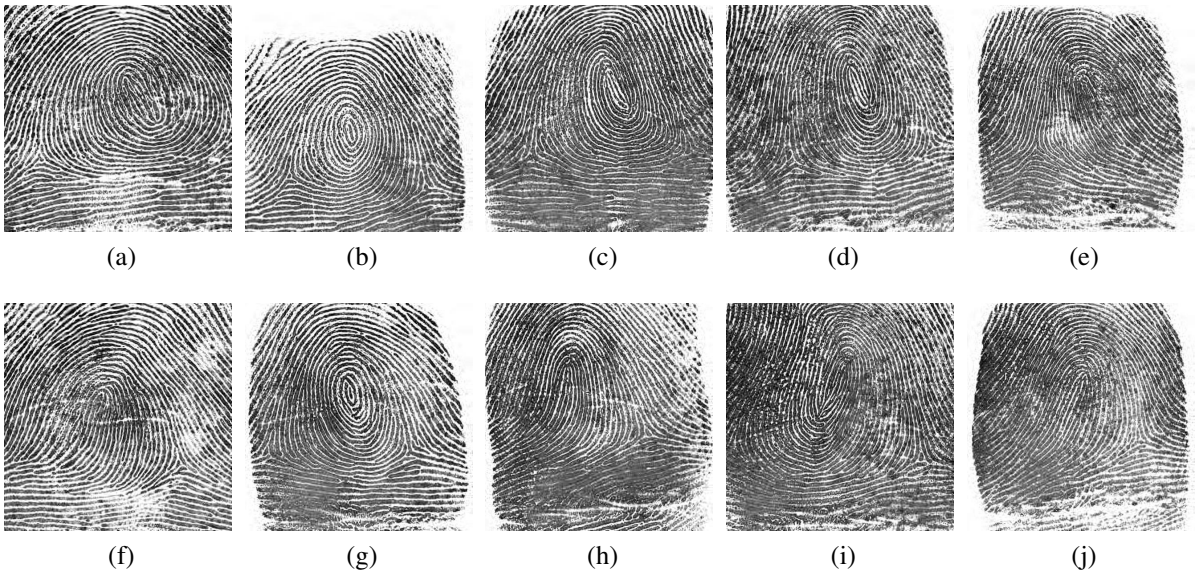

Figura 19 – Impressões digitais utilizados no experimento I: (a) Polegar Direito, (b) Indicador Direito, (c) Médio Direito, (d) Anular Direito, (e) Mínimo Direito, (f) Polegar Esquerdo, (g) Indicador Esquerdo, (h) Médio Esquerdo, (i) Anular Esquerdo, (j) Mínimo Esquerdo.
Fonte: Autoria própria.

Todas as pesquisas foram identificadas, ou seja, o polegar direito é comparado apenas com polegares direitos. Da mesma forma, o indicador direito é comparado apenas com indicadores direitos etc. Logo, nas pesquisas contendo 1 dedo, aproximadamente 10% dos *templates* são visitados e na pesquisa contendo a decadatilar, 100% dos *templates* são visitados. O número máximo de candidatos retornados neste experimento foi configurado para 10.

Neste cenário, o ABIS é utilizado para pesquisas rápidas, como por exemplo, em delegacias, barreiras policiais ou ainda para evitar o cadastro de duplicados ou fraudes. Como o tempo de resposta se torna um elemento importante, foram utilizados dois algoritmos encadeados, conforme exposto anteriormente.

A Figura 20 apresenta os tempos de identificação das pesquisas individuais e a Figura 21 os tempos de identificação da pesquisa decadatilar. A Figura 22 apresenta os ganhos de velocidade obtidos com a adição de nós na topologia. A Tabela 1 apresenta todos os tempos de identificação obtidos e os respectivos ganhos de velocidade.

Nos cálculos de ganho de velocidade, a topologia contendo 3 nós foi considerada como base, pois é a topologia mínima necessária para que todos os *templates* sejam carregados na memória.

Através do gráfico da Figura 22 é possível observar que a pesquisa decadatilar foi a que obteve maiores ganhos de velocidade com o aumento do número de nós na topologia. Pode-se

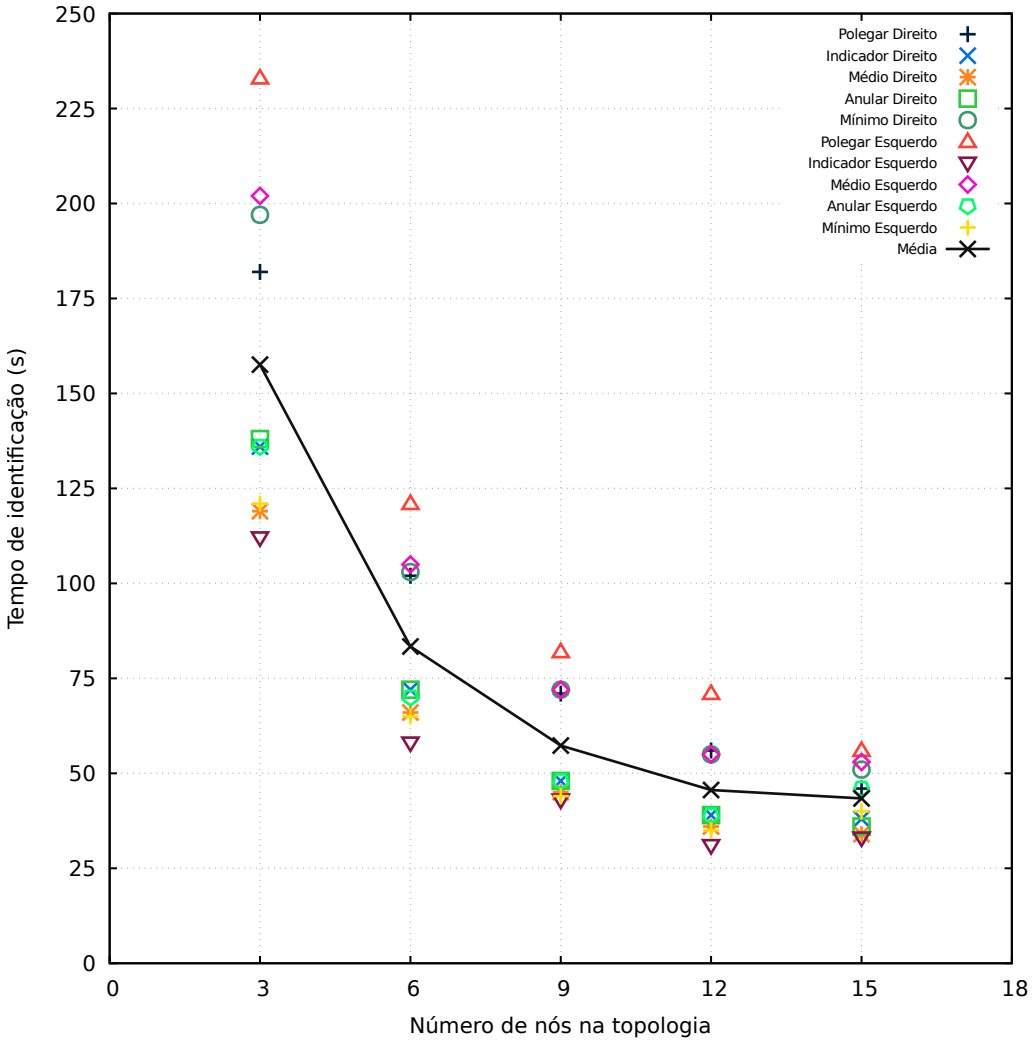

Figura 20 – Tempos de resposta de 10 identificações comuns, contendo uma impressão digital, contra registros civis. A variação no tempo de processamento ocorre devido ao número de minúcias e ao tamanho do conjunto de candidatos produzidos pelo algoritmo de confronto rápido.
Fonte: Autoria própria

constatar ainda que algumas identificações sofreram perda de performance na topologia de 15 nós.

Considerando a topologia de base (3 nós), os mais de 5 milhões de registros decadatilares foram divididos em 240 partições (3×80). Cada partição foi processada por uma subtarefa, em uma *thread*, produzindo uma lista contendo no máximo 10 candidatos. Desta forma, o conjunto de candidatos processados pelo controlador continha aproximadamente 2.400 elementos. O tamanho médio de um objeto de similaridade (candidato) é de aproximadamente 100 KB. Logo, na topologia de base existe um tráfego de aproximadamente 234 MB entre os nós de processamento e o controlador.

Analisando agora a topologia com 15 nós, os registros foram divididos em 1.200 partições (15×80). Logo, a cardinalidade do conjunto de candidatos processados pelo controlador era de aproximadamente 12.000. Desta forma, nesta topologia existe um tráfego de aproximadamente

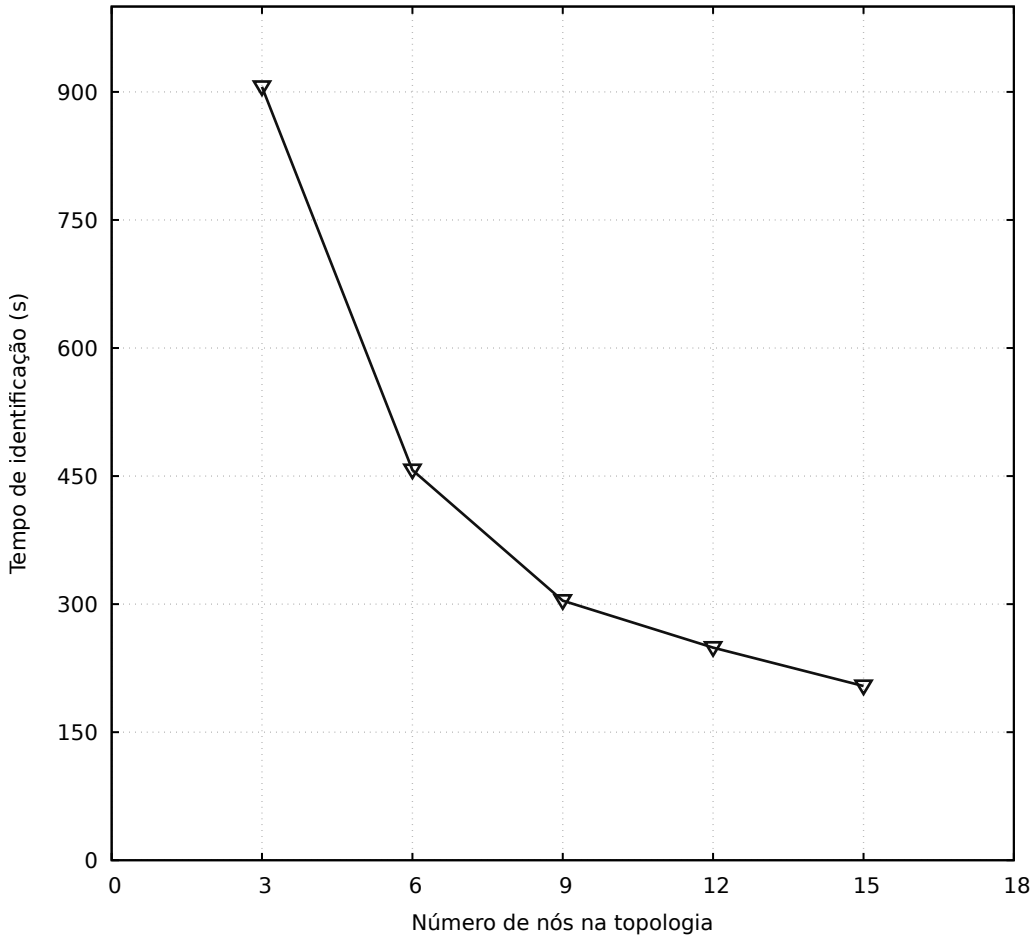

Figura 21 – Tempos de resposta de uma pesquisa comum, contendo 10 impressões digitais, contra registros civis.

Fonte: Autoria própria

1,14 GB entre os nós e o controlador. Como o tempo de processamento da ficha decadatilar é aproximadamente 5 vezes maior do que das demais pesquisas realizadas (Tabela 1), os tempos de envio dos resultados das subtarefas e o tempo de processamento destes pelo controlador não afetam significativamente a escalabilidade do sistema. Por outro lado, como as pesquisas individuais são mais rápidas, elas acabam sendo penalizadas por uma topologia com muitos nós, onde os tempos de comunicação são maiores do que os tempos de processamento.

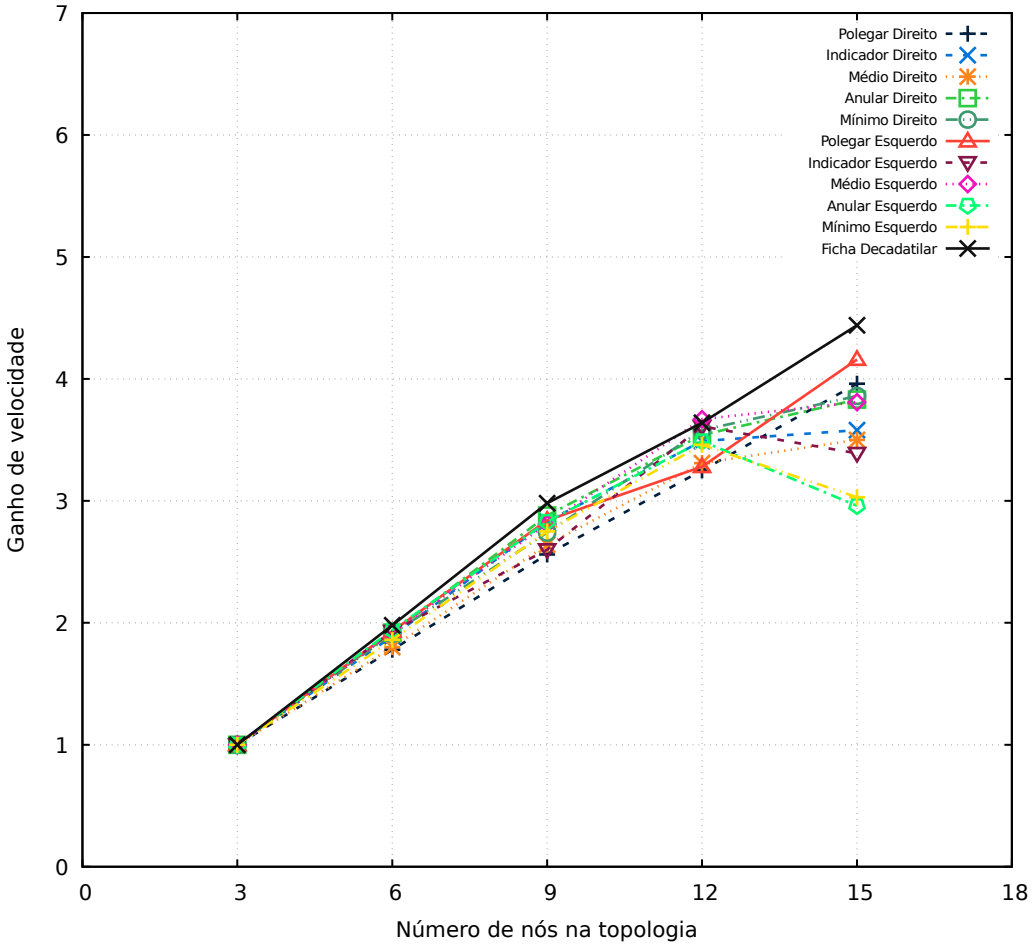

Figura 22 – Ganhos de velocidade de identificações comuns contra registros civis.
Fonte: Autoria própria

Tabela 1 – Tempos de resposta e ganhos de velocidade de identificações comuns contra registros civis.

Número de Nós	Dedo	Tempo (s)	Ganho de velocidade
3	Polegar Direito	182	1,00
	Indicador Direito	136	1,00
	Médio Direito	119	1,00
	Anelar Direito	138	1,00
	Mínimo Direito	197	1,00
	Polegar Esquerdo	233	1,00
	Indicador Esquerdo	112	1,00
	Médio Esquerdo	202	1,00
	Anelar Esquerdo	136	1,00
	Mínimo Esquerdo	121	1,00
	Ficha Decadatilar	906	1,00
6	Polegar Direito	102	1,78
	Indicador Direito	72	1,89
	Médio Direito	66	1,80
	Anelar Direito	72	1,92
	Mínimo Direito	103	1,91
	Polegar Esquerdo	121	1,93
	Indicador Esquerdo	58	1,93
	Médio Esquerdo	105	1,92
	Anelar Esquerdo	70	1,94
	Mínimo Esquerdo	65	1,86
	Ficha Decadatilar	457	**1,98**
9	Polegar Direito	71	2,56
	Indicador Direito	48	2,83
	Médio Direito	45	2,64
	Anelar Direito	48	2,88
	Mínimo Direito	72	2,74
	Polegar Esquerdo	82	2,84
	Indicador Esquerdo	43	2,60
	Médio Esquerdo	72	2,81
	Anelar Esquerdo	48	2,83
	Mínimo Esquerdo	44	2,75
	Ficha Decadatilar	304	**2,98**
12	Polegar Direito	56	3,25
	Indicador Direito	39	3,49
	Médio Direito	36	3,31
	Anelar Direito	39	3,54
	Mínimo Direito	55	3,58
	Polegar Esquerdo	71	3,28
	Indicador Esquerdo	31	3,61
	Médio Esquerdo	55	**3,67**
	Anelar Esquerdo	39	3,49
	Mínimo Esquerdo	35	3,46
	Ficha Decadatilar	249	3,64
15	Polegar Direito	46	3,96
	Indicador Direito	38	3,58
	Médio Direito	34	3,50
	Anelar Direito	36	3,83
	Mínimo Direito	51	3,86
	Polegar Esquerdo	56	4,16
	Indicador Esquerdo	33	3,39
	Médio Esquerdo	53	3,81
	Anelar Esquerdo	46	2,96
	Mínimo Esquerdo	40	3,03
	Ficha Decadatilar	204	**4,44**

Fonte: Autoria própria

5.2.2 Experimento II

Neste experimento, foram alocados até 15 nós para o *cluster* de processamento e as topologias avaliadas foram as mesmas do experimento I.

Um fragmento de impressão digital (Figura 23) contendo 24 minúcias foi confrontado contra os registros civis. Esta latente pertence a um indivíduo do sexo masculino, com idade entre 25 e 35 anos e é proveniente do anular direito.

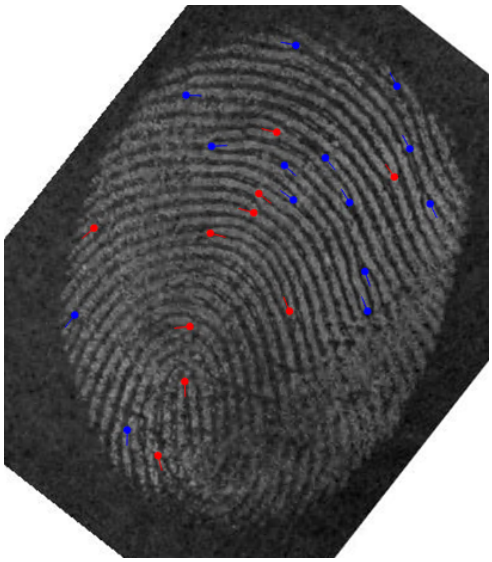

Figura 23 – Fragmento de impressão digital utilizado no experimento II com 24 minúcias. Fonte: Autoria própria.

Foram enviadas 7 identificações para cada topologia avaliada, cada uma utilizando diferentes critérios de redução do espaço de busca: sem filtro; fgp; sexo; faixa etária; fgp e sexo; fgp e faixa etária; e fgp, sexo e faixa etária. O número máximo de candidatos retornados neste experimento foi configurado para 100. Este número deve ser configurado de acordo com as particularidades de cada sistema como: qualidade das imagens armazenadas, algoritmo de confronto utilizado e experiência dos peritos papiloscopistas.

Neste tipo de cenário, o ABIS é utilizado principalmente para auxiliar na análise de cenas de crime e identificação *post-mortem*. O fator mais importante nestes casos é a precisão do algoritmo de confronto do que o tempo de identificação. Sendo assim, foi utilizado somente o algoritmo de confronto mais preciso, porém mais lento.

A Figura 24 apresenta os tempos de identificação resultantes deste experimento. A Figura 25 mostra os ganhos de velocidade obtidos com a variação do número de nós na topologia. A Tabela 2 contém os valores dos tempos de identificação e dos ganhos de velocidade.

Os cálculos dos ganhos de velocidade seguiram o mesmo esquema do experimento I. Através do gráfico da Figura 25 é possível constatar que quase todas as pesquisas obtiveram ganhos de velocidade quando o número de nós da topologia foi aumentado. Porém, duas delas (Fgp + Idade e Fgp + Sexo + Idade) sofreram queda de desempenho na topologia de 15 nós.

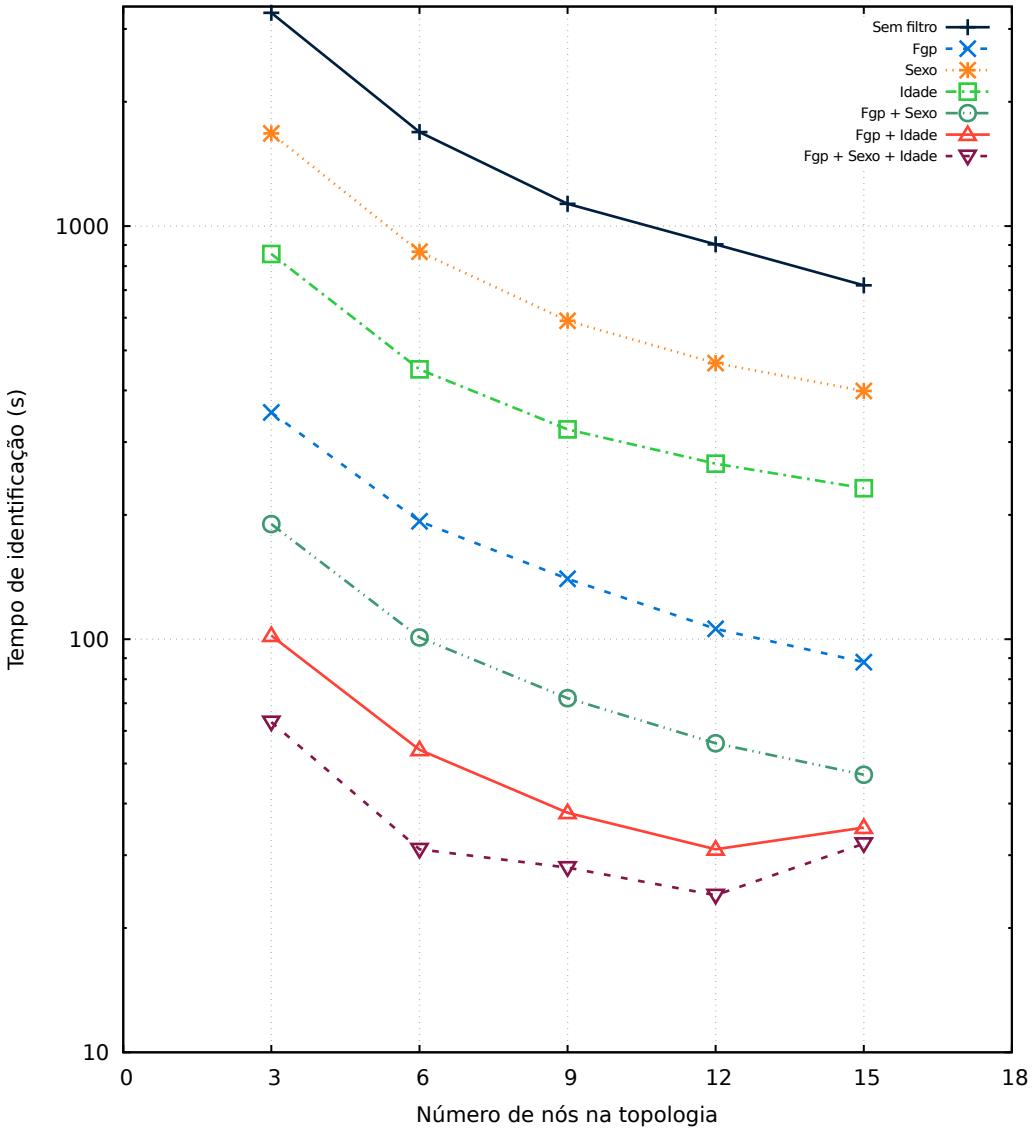

Figura 24 – Tempos de resposta de identificações forenses contra registros civis.
Fonte: Autoria própria

Os motivos desta queda são os mesmos apresentados na análise do experimento I. Na topologia de base, o conjunto de candidatos processados pelo controlador era de aproximadamente 24.000 elementos ($3 \times 80 \times 100$). Considerando que o tamanho médio de um objeto de similaridade é de aproximadamente 100 KB, existe um tráfego de aproximadamente 2,29 GB entre os nós de processamento e o controlador.

Na topologia com 15 nós, o conjunto de candidatos analisados pelo controlador era de aproximadamente 120.000 elementos ($15 \times 80 \times 100$). Sendo assim, existe um tráfego de aproximadamente 11.44 GB entre os nós de processamento e o controlador. Logo, a pesquisa sem filtros sofre menor penalização pois o tempo de processamento é muito maior que os tempos de comunicação. Por outro lado, na pesquisa com todos os filtros (Fgp + Sexo + Idade), que possui um tempo de processamento 22 vezes menor (Tabela 2) do que a pesquisa sem filtros, os tempos de comunicação são maiores que os tempos de processamento, ocasionando uma maior

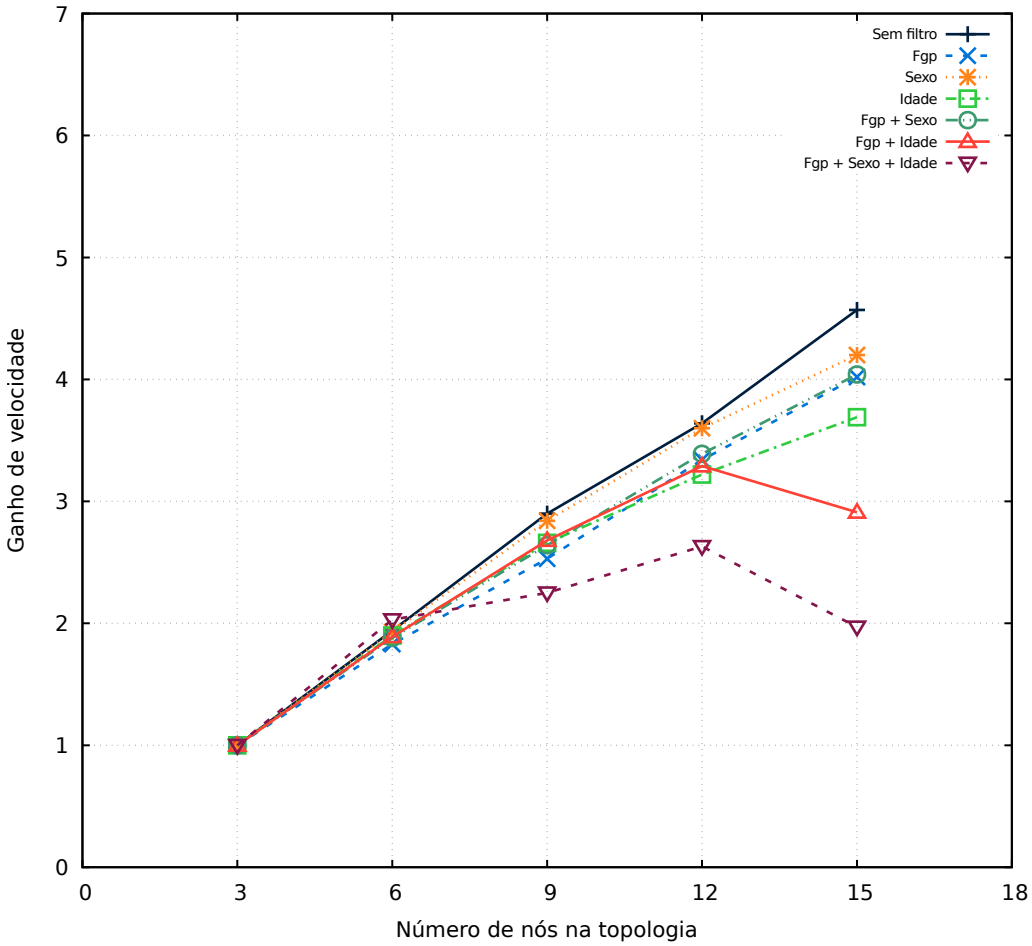

Figura 25 – Ganhos de velocidade de identificações forenses contra registros civis. Fonte: Autoria própria

penalização.

Tabela 2 – Tempos de resposta e ganhos de velocidade de identificações forenses contra registros civis.

Número de Nós	Critério de Penetração	Tempo (s)	Ganho de velocidade
3	Sem filtro	3283	1,00
	Fgp	354	1,00
	Sexo	1677	1,00
	Idade	856	1,00
	Fgp + Sexo	190	1,00
	Fgp + Idade	102	1,00
	Fgp + Sexo + Idade	63	1,00
6	Sem filtro	1689	1,94
	Fgp	193	1,83
	Sexo	867	1.93
	Idade	450	1,90
	Fgp + Sexo	101	1,88
	Fgp + Idade	54	1,89
	Fgp + Sexo + Idade	31	**2.03**
9	Sem filtro	1132	**2,90**
	Fgp	140	2,53
	Sexo	590	2,84
	Idade	322	2,66
	Fgp + Sexo	72	2,64
	Fgp + Idade	38	2,68
	Fgp + Sexo + Idade	28	2,25
12	Sem filtro	903	**3,64**
	Fgp	106	3,34
	Sexo	466	3,60
	Idade	266	3,22
	Fgp + Sexo	56	3,39
	Fgp + Idade	31	3,29
	Fgp + Sexo + Idade	24	2,63
15	Sem filtro	719	**4,57**
	Fgp	88	4,02
	Sexo	399	4,20
	Idade	232	3,69
	Fgp + Sexo	47	4,04
	Fgp + Idade	35	2,91
	Fgp + Sexo + Idade	32	1,97

Fonte: Autoria própria

5.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste capítulo foram apresentados alguns experimentos que validaram a solução proposta. Cabe ressaltar que o objetivo principal deste trabalho não é avaliar o tempo de identificação do sistema, mas sim sua escalabilidade. Entretanto, considerando o cenário forense, os tempos de resposta do sistema são adequados, visto que geralmente o prazo máximo para processamento de uma pesquisa de latente é de 24 horas (ACRE, 2016; RIO DE JANEIRO, 2008).

Uma base de dados contendo mais de 50 milhões de impressões digitais foi utilizada e os experimentos foram realizados em um ambiente real de produção, para que o comportamento do sistema fosse avaliado da melhor forma possível.

No primeiro experimento, um encadeamento de algoritmos foi utilizado para reduzir o tempo de identificação, simulando um cenário de pesquisas rápidas em delegacias e barreiras policiais. Foram obtidos ganhos de velocidade de até 4,44 vezes quando utilizados 5 vezes mais nós do que a topologia base. Para o dobro e o triplo de nós, os ganhos máximos foram praticamente lineares, de 1,98 e 2,98, respectivamente.

No segundo experimento, apenas o algoritmo preciso foi utilizado, simulando um cenário de pesquisas forenses. Foram obtidos ganhos de velocidade de até 4,57 vezes, quando utilizados 5 vezes mais nós do que a topologia base. Para o dobro de nós, o maior ganho obtido foi de 2,03.

Em ambos os experimentos ocorreram casos em que o ganho de velocidade diminuiu com o aumento da topologia. Quanto menor o número de elementos visitados pelo algoritmo preciso em uma partição (devido a critérios de penetração ou a algoritmos rápidos de confronto) e maior o poder computacional disponível, o tempo de processamento torna-se menor do que o tempo de comunicação, logo não se obtém ganhos. Além disso, quanto maior a topologia, maior o número de candidatos que devem ser processados pelo controlador para produzir o conjunto final de candidatos.

6 CONCLUSÕES

Considerando a necessidade cada vez maior de autenticar e identificar pessoas utilizando biometria, o grande volume de dados envolvidos e a complexidade de tais procedimentos, é preciso criar novos meios para realizar estas tarefas.

Neste trabalho foi proposta uma abordagem de implementação escalável para uma central de confrontos de um sistema automático de identificação biométrica. Para tanto, foi feita uma revisão da literatura sobre os assuntos abordados, bem como dos trabalhos relacionados. A partir disso, projetou-se a arquitetura do sistema para que atendesse aos principais casos de uso de um ABIS: identificação civil, criminal e forense.

A solução proposta foi implementada utilizando o Apache Ignite, um *framework* para computação em memória, que permite manipular um grande volume de dados em tempo real e faz com que a aplicação tenha escalabilidade horizontal.

Para validar a solução proposta, foram realizados dois experimentos utilizando até 16 nós, sendo 15 de armazenamento e processamento e 1 de controle. A base de dados utilizada continha mais de 50 milhões de impressões digitais capturadas. A central de confrontos foi avaliada em um cenário real de produção, juntamente com os demais componentes do sistema. Nestes experimentos foram submetidas diversas identificações comuns e forenses em 5 topologias diferentes para avaliar os ganhos de velocidade obtidos com o aumento do número de nós no *cluster* e a capacidade do sistema de operar com milhões de registros biométricos.

Foram observados em alguns casos ganhos de velocidade praticamente lineares com a inclusão de nós na topologia. Em outros porém, não foram observados ganhos. Isto pode ser explicado pela diminuição expressiva da carga de processamento quando são aplicados filtros para redução do espaço de busca, fazendo com que os tempos de comunicação penalizem os resultados obtidos.

Há pontos para continuidade deste trabalho, como a migração desta solução para computação em nuvem utilizando *clouds* seguras (Secure Cloud, 2017), otimização do envio dos resultados para reduzir os tempos de comunicação e evitar a degradação de performance com o aumento da topologia, adicionar um módulo de supervisão, implementar os algoritmos de confronto em Java para reduzir os tempos gastos no envio de dados entre o Java e o C++ e realizar mais experimentos com outros tamanhos de bases e outros algoritmos de confronto. Estes experimentos não foram realizados devido a limitações de uso no *cluster* utilizado.

REFERÊNCIAS

ACRE. *Edital do Pregão Presencial para Registro de Preços nº. 450/2016 – CEL 01*. 2016. Disponível em: <http://www.licitacao.ac.gov.br/cpl/sie/arquivos/editais/PREGAO%20PRESENCIAL%20SRP%20N%20450%202016%20CEL%2001%20SEPC%20CONT%20DE%20EMP%20IMPLATACAO%20DA%20SOLUCAO%20INTEGRADA.pdf>. Citado na página 59.

Antheus Tecnologia Ltda. *Soluções Biométricas de Identificação*. 2017. Disponível em: <http://www.antheus.com.br>. Citado na página 48.

BERTILLON, A. *Identification anthropométrique*. Melun: Imprimerie Administrative Melun, 1893. 384 p. Disponível em: <https://archive.org/details/identificationan00bert>. Citado na página 19.

BEY, K. B. et al. Agent based approach for distribution of fingerprint matching in a metacomputing environment. In: *Proceedings of the 8th International Conference on New Technologies in Distributed Systems*. New York, NY, USA: ACM, 2008. (NOTERE '08), p. 29:1–29:7. ISBN 978-1-59593-937-1. Disponível em: <http://doi.acm.org/10.1145/1416729.1416766>. Citado 3 vezes nas páginas 9, 30 e 31.

BHUIYAN, S. A.; ZHELUDKOV, M.; ISACHENKO, T. *High Performance in-memory computing with Apache Ignite: building low latency, near real time application*. [s.n.], 2016. Disponível em: <https://leanpub.com/ignite>. Citado 2 vezes nas páginas 25 e 27.

CAMPISI, P. et al. *IEEE Certified Biometrics Professional (CBP) Learning System : Module 1 Biometrics Fundamentals*. Eagan, MN: IEEE, 2010. Citado 2 vezes nas páginas 18 e 19.

CAMPISI, P. et al. *IEEE Certified Biometrics Professional (CBP) Learning System : Module 3 Biometric System Design and Evaluation*. Eagan, MN: IEEE, 2010. Citado na página 25.

CAMPISI, P. et al. *IEEE Certified Biometrics Professional (CBP) Learning System : Module 6 Biometrics Applications*. Eagan, MN: IEEE, 2010. Citado 2 vezes nas páginas 21 e 23.

CAO, K.; JAIN, A. K. Learning fingerprint reconstruction: From minutiae to image. *IEEE Transactions on Information Forensics and Security*, Institute of Electrical and Electronics Engineers (IEEE), v. 10, n. 1, p. 104–117, jan 2015. Disponível em: <https://doi.org/10.1109/tifs.2014.2363951>. Citado na página 31.

CAPPELLI, R.; FERRARA, M.; MALTONI, D. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 32, n. 12, p. 2128–2141, dez. 2010. ISSN 0162-8828. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2010.52>. Citado 2 vezes nas páginas 35 e 36.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, ACM, New York, NY, USA, v. 51, n. 1, p. 107–113, jan. 2008. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1327452.1327492>. Citado 2 vezes nas páginas 28 e 44.

Docker. *Docker: Build, ship, and Run Any App, Anywhere*. 2017. Disponível em: <https://www.docker.com/>. Citado na página 26.

FBI. *Next Generation Identification (NGI)*. 2016. Disponível em: <https://www.fbi.gov/services/cjis/fingerprints-and-other-biometrics/ngi>. Citado na página 24.

FBI. *Next Generation Identification (NGI) Monthly Fact Sheet : October 2016 Monthly Statistics*. 2016. Disponível em: <https://www.fbi.gov/file-repository/ngi-monthly-fact-sheet>. Citado na página 24.

FENG, J.; JAIN, A. K. Fingerprint reconstruction: From minutiae to phase. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 33, n. 2, p. 209–223, feb 2011. Disponível em: <https://doi.org/10.1109/tpami.2010.77>. Citado na página 31.

GOTTSCHLICH, C.; HUCKEMANN, S. Separating the real from the synthetic: minutiae histograms as fingerprints of fingerprints. *IET Biometrics*, Institution of Engineering and Technology (IET), v. 3, n. 4, p. 291–301, dec 2014. Disponível em: <https://doi.org/10.1049/iet-bmt.2013.0065>. Citado na página 36.

GridGain Systems. *GridGain vs Hazelcast Benchmarks*. 2017. Disponível em: <https://www.gridgain.com/resources/benchmarks/gridgain-vs-hazelcast-benchmarks>. Citado na página 28.

GridGrain Systems. Introducing apache ignite: A gridgain systems in-memory computing. 2017. Citado 2 vezes nas páginas 27 e 28.

Hazelcast. *The Leading Open Source In-Memory Data Grid: Distributed Computing, Simplified.* 2017. Disponível em: <https://hazelcast.org/>. Citado na página 26.

JAIN, A.; HONG, L.; BOLLE, R. On-line fingerprint verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 19, n. 4, p. 302–314, apr 1997. Disponível em: <https://doi.org/10.1109/34.587996>. Citado na página 21.

JAIN, A. K.; FLYNN, P.; ROSS, A. A. (Ed.). *Handbook of Biometrics*. New York, NY: Springer, 2008. Citado 4 vezes nas páginas 16, 21, 22 e 23.

JIANG, X.; YAU, W.-Y. Fingerprint minutiae matching based on the local and global structures. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. IEEE Comput. Soc, 2000. Disponível em: <https://doi.org/10.1109/icpr.2000.906252>. Citado 2 vezes nas páginas 35 e 36.

JOHNS, M. *Getting Started with Hazelcast*. Packt Publishing, 2013. Disponível em: <https://www.amazon.com/Getting-Started-Hazelcast-Mat-Johns-ebook/dp/B00EUA1X4S?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00EUA1X4S>. Citado na página 26.

KOMARINSKI, P. *Automated Fingerprint Identification Systems (AFIS)*. Burlington, MA: Academic Press, 2005. Citado 2 vezes nas páginas 18 e 19.

KRICHEN, E. Connecting a billion and more: focus on iris recognition. XIII International Summer School on Biometrics. 2016. Citado na página 24.

LUCK, G. *Mastering Hazelcast IMDG Online Book*. 2017. Disponível em: <https://hazelcast.org/mastering-hazelcast>. Citado na página 26.

MALTONI, D. Generation of synthetic fingerprint image databases. In: *Automatic Fingerprint Recognition Systems*. Springer-Verlag, 2004. p. 361–384. Disponível em: <https://doi.org/10.1007/0-387-21685-5_18>. Citado na página 34.

MALTONI, D. et al. *Handbook of Fingerprint Recognition*. Second. London: Springer, 2009. Citado 5 vezes nas páginas 16, 19, 20, 37 e 38.

MATHEW, R.; THOMAS, B.; KIZHAKKETHOTTAM, J. J. Review on latent fingerprint matching techniques. In: *2015 International Conference on Soft-Computing and Networks Security (ICSNS)*. IEEE, 2015. Disponível em: <https://doi.org/10.1109/icsns.2015.7292415>. Citado na página 43.

MIRON, T. S. L. R. F.; HULEA, M. Two server topologies for a distributed fingerprint-based recognition system. *15th International Conference on System Theory, Control, and Computing (ICSTCC)*, 2011. Citado 4 vezes nas páginas 9, 31, 32 e 33.

National Science and Technology Council. *Biometrics Glossary*. Subcommittee on Biometrics, 2006. Disponível em: <http://www.biometrics.gov/Documents/Glossary.pdf>. Citado na página 16.

PERALTA, D. et al. Minutiae-based fingerprint matching decomposition: Methodology for big data frameworks. *Information Sciences*, Elsevier BV, v. 408, p. 198–212, oct 2017. Disponível em: <https://doi.org/10.1016/j.ins.2017.05.001>. Citado 3 vezes nas páginas 9, 36 e 37.

PERALTA, D. et al. DPD-DFF: A dual phase distributed scheme with double fingerprint fusion for fast and accurate identification in large databases. *Information Fusion*, Elsevier BV, v. 32, p. 40–51, nov 2016. Disponível em: <https://doi.org/10.1016/j.inffus.2016.03.002>. Citado 6 vezes nas páginas 9, 16, 35, 36, 37 e 44.

PERALTA, D. et al. Fast fingerprint identification for large databases. *Pattern Recognition*, Elsevier BV, v. 47, n. 2, p. 588–602, feb 2014. Disponível em: <https://doi.org/10.1016/j.patcog.2013.08.002>. Citado 10 vezes nas páginas 9, 21, 22, 30, 32, 33, 34, 35, 36 e 37.

Pivotal. *RabbitMQ*. 2017. Disponível em: <https://www.rabbitmq.com/>. Citado na página 48.

RATHA, N. K.; CONNELL, J. H.; PANKANTI, S. Big data approach to biometric-based identity analytics. v. 59, n. 2/3, 2015. Citado 6 vezes nas páginas 16, 21, 22, 23, 24 e 25.

Red Hat. *Infinispan 9.1 User Guide*. 2017. Disponível em: <http://infinispan.org/docs/stable/user_guide/user_guide.html>. Citado 2 vezes nas páginas 26 e 27.

Red Hat. *Introduction to Infinispan: Distributed in-memory key/value data grid and cache*. 2017. Disponível em: <http://infinispan.org/about/>. Citado 2 vezes nas páginas 26 e 27.

RIO DE JANEIRO. *Anexo 6: Projeto Básico: Sistema Estadual de Identificação – SEI*. 2008. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=17&ved=0ahUKEwjUh4-x19LWAhWEFZAKHfxBCcQ4ChAWCEkwBg&url=http%3A%2F%2Fwww.detran.rj.gov.br%2F_aplicacoes%2F_aplicacoes_cadastro_edital_download_blob7.asp%3Fid%3D150&usg=AOvVaw09Y-N4lMxPZLR3WSagsOKi>. Citado na página 59.

ROSS, A.; SHAH, J.; JAIN, A. K. From template to image: Reconstructing fingerprints from minutiae points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 29, n. 4, p. 544–560, apr 2007. Disponível em: <https://doi.org/10.1109/tpami.2007.1018>. Citado na página 31.

Safran Identity & Security. *Aadhaar: a unique ID number for all*. 2016. Disponível em: <http://www.morpho.com/en/media/20151123_aadhaar-unique-id-number-all>. Citado na página 24.

SANKARAN, A.; VATSA, M.; SINGH, R. Latent fingerprint matching: A survey. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 2, p. 982–1004, 2014. Disponível em: <https://doi.org/10.1109/access.2014.2349879>. Citado na página 43.

Secure Cloud. *Secure Big Data Processing in Untrusted Clouds*. 2017. Disponível em: <https://www.securecloudproject.eu/>. Citado na página 60.

SETRAKYAN, D. *Emergence of the In-Memory Data Fabric*. 2015. Disponível em: <https://dzone.com/articles/emergence-memory-data-fabric>. Citado 2 vezes nas páginas 25 e 26.

THALER, D.; RAVISHANKAR, C. Using name-based mappings to increase hit rates. *IEEE/ACM Transactions on Networking*, Institute of Electrical and Electronics Engineers (IEEE), v. 6, n. 1, p. 1–14, 1998. Disponível em: <https://doi.org/10.1109/90.663936>. Citado na página 43.

The Apache Software Foundation. *Apache Hadoop*. 2017. Disponível em: <http://hadoop.apache.org/>. Citado na página 36.

The Apache Software Foundation. *Apache Ignite*. 2017. Disponível em: <https://ignite.apache.org/>. Citado na página 26.

The Apache Software Foundation. *Apache Ignite: Technical Documentation*. 2017. Disponível em: <https://apacheignite.readme.io/docs>. Citado na página 43.

The Apache Software Foundation. *Apache Spark*. 2017. Disponível em: <https://spark.apache.org/>. Citado na página 36.

WATSON, C. *NIST 8-Bit Gray Scale Images of Fingerprint Image Groups (FIGS), NIST Special Database 4*. NIST, 1992. Disponível em: <http://www.nist.gov/srd/nistsd4.cfm>. Citado na página 34.

WATSON, C. *NIST Mated Fingerprint Card Pairs 2 (MFCP2), NIST Special Database 14*. National Institute of Standards and Technology, 1993. Disponível em: <http://www.nist.gov/srd/nistsd14.cfm>. Citado na página 34.