UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

WAGNER SENGER

ARQUITETURA PARA CLUSTERIZAÇÃO DE RECURSOS BASEADO EM SEU PODER COMPUTACIONAL UTILIZANDO ALGORITMO HIERÁRQUICO E ASSINATURA COMPORTAMENTAL

DISSERTAÇÃO

PONTA GROSSA

2018

WAGNER SENGER

ARQUITETURA PARA CLUSTERIZAÇÃO DE RECURSOS BASEADO EM SEU PODER COMPUTACIONAL UTILIZANDO ALGORITMO HIERÁRQUICO E ASSINATURA COMPORTAMENTAL

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação, da Diretoria de Pesquisa e Pós-Graduação, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Lourival Aparecido de Góis

PONTA GROSSA
2018

Ficha catalográfica elaborada pelo Departamento de Biblioteca
da Universidade Tecnológica Federal do Paraná, Campus Ponta Grossa
n.58/18

S476 Senger, Wagner

Arquitetura para clusterização de recursos baseado em seu poder computacional utilizando algoritmo hierárquico e assinatura comportamental. / Wagner Senger, 2018.
77 f. : il. ; 30 cm.

Orientador: Prof. Dr. Lourival Aparecido de Góis

Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação. Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018.

1. Cluster (Sistema de computador). 2. Assinaturas digitais. 3. Algorítmos computacionais. 4. Sistemas de computação em grade. I. Góis, Lourival Aparecido de. II. Universidade Tecnológica Federal do Paraná. III. Título.

CDD 004

Elson Heraldo Ribeiro Junior. CRB-9/1413. 28/11/2018.

FOLHA DE APROVAÇÃO

Título de Dissertação Nº 6/2018

ARQUITETURA PARA CLUSTERIZAÇÃO DE RECURSOS BASEADO EM SEU PODER COMPUTACIONAL UTILIZANDO ALGORITMO HIERÁRQUICO E ASSINATURA COMPORTAMENTAL

Por

Wagner Senger

Esta dissertação foi apresentada às **14 horas** de **05 Novembro 2018**, na sala da DIRPPG, Bloco E, como requisito parcial para a obtenção do título de MESTRE EM CIÊNCIA DA COMPUTAÇÃO, Programa de Pós-Graduação em Ciência da Computação. O candidato foi arguido pela Banca Examinadora, composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Prof. Dr. Augusto Foronda (UTFPR)

Prof. Dr. Rodolfo Miranda de Barros (UEL)

Prof. Dr. Lourival Aparecido de Góis(UTFPR)
Orientador e presidente da banca

Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Visto da Coordenadora:

Profª. Drª. Sheila Morais de Almeida
Coordenadora do PPGCC
UTFPR – Câmpus Ponta Grossa

A Folha de Aprovação assinada encontra-se arquivada na Secretaria Acadêmica

Dedico esta tese aos meus pais, que
com muito sacrifício me possibilitaram
o acesso aos estudos.

AGRADECIMENTOS

Agradeço:

- Primeiramente a Deus, por ter permitido que as coisas acontecessem no tempo correto para que este mestrado fosse possível.
- A minha família, em especial aos meus pais que me incentivaram a continuar buscando por conhecimento, e nunca mediram esforços para me proporcioná-lo.
- A minha namorada Heloisa Pereira Rodrigues, que sempre me apoiou, mesmo quando eu não acreditei que era possível, sempre compreendendo minhas ausências e períodos de dedicação, sendo um porto seguro para meus momentos de insegurança.
- Ao meu orientador Lourival Aparecido de Góis, professor e amigo, que desde minha graduação, com paciência sempre se dispôs a me auxiliar, incentivar e contribuir para meu crescimento pessoal, sem contar as longas horas dedicadas a propor, corrigir e orientar este trabalho.
- Aos professores do PPGCC, que se esforçaram para nos proporcionar em todas as ocasiões o melhor ensino possível.
- Aos professores Augusto Foronda e Rodolfo Miranda de Barros, membros da banca, que se disponibilizaram a auxiliar no crescimento deste trabalho com avaliações, correções e sugestões.
- Aos meus colegas de turma, Renê Pomilio de Oliveira, Helbert da Rocha, Felipe Bueno, Lucas Fernando Souza de Castro, Luan Bukowitz Beluzzo, Mayara Midori Omai, Thissiany Beatriz Almeida e Víctor Pedroso Ambiel Barros, que compartilharam as angustias e dividiram por tantas vezes o fardo de disciplinas pesadas e complexas.
- A UTFPR, que mais uma vez me aceitou como seu aluno.

“A verdadeira viagem de descobrimento
não consiste em procurar novas paisagens,
mas em ter novos olhos.“
Marcel Proust

RESUMO

SENGER, Wagner. **Arquitetura para clusterização de recursos baseado em seu poder computacional utilizando Algoritmo Hierárquico e Assinatura Comportamental.** 2018. Dissertação (Mestrado em Ciência da Computação) - Universidade Tecnológica Federal Do Paraná. Ponta Grossa, 2018.

Em uma Grade Computacional, recursos com baixo poder computacional, ao concorrer diretamente com todos em um ambiente podem vir a ser subutilizados, causando com o tempo a sua fuga. Para mitigar este efeito, e proporcionar um melhor uso dos recursos do ambiente, este estudo propõe uma arquitetura que se baseia na organização dos elementos em grupos, e que para isso, seus componentes internos como um todo sejam levadas em consideração, e não apenas a capacidade da sua CPU. Para tal, a quantidade de cada componente relevante dos recursos para a aplicação alvo é compilada, compondo o seu poder computacional. Os recursos são dispostos em grupos, cuja soma do poder computacional de seus integrantes é aproximada dos demais. A separação dos recursos em grupos é executada por um algoritmo hierárquico com base na maior distância entre os valores, devido ao equilíbrio entre elementos fortes e fracos que ele proporciona, gerando assim grupos com valor total aproximado. Esta organização permite que quando houver a necessidade de definição de qual grupo será o responsável pela execução de uma tarefa, qualquer um tenha capacidade de atendê-la, visto que o poder de processamento não é mais um determinante para tal. Para que exista um parâmetro de escolha do grupo que será selecionado para execução, o comportamento padrão dos recursos é analisado. Cada recurso possui um padrão de utilização, que pode ser identificado através do seu acompanhamento, a ponto de que sua utilização seja previsível. Este padrão de utilização é representado por um perfil de comportamento denominado Assinatura Comportamental. Após determinada a assinatura de cada recurso e também dos grupos aos quais pertencem, como todos estes grupos passam a ter condições similares de atender uma demanda, a assinatura permite a sua distinção, retratando o momento mais oportuno de utilização de cada, proporcionando uma métrica de escolha do destino da requisição. Neste estudo são implementados os métodos propostos para a clusterização, Assinatura Comportamental e a Assinatura dos *clusters*. O escalonamento das tarefas não é implementado, visto que um aprofundamento específico nesta área é necessário, portanto comparar o número de tarefas recebidas por um recurso neste método, com métodos tradicionais onde o recurso concorre diretamente com outros não é possível. É possível porém determinar os resultados da implantação dos métodos, e se a arquitetura permitiu a criação de um ambiente tal qual proposto.

Palavras-chaves: *Cluster*. Assinatura Comportamental. Balanceamento de Carga. Algoritmo Hierárquico. *Grid*.

ABSTRACT

SENGER, Wagner. **Computational power resource based clustering architecture using Hierarchical Algorithm and Behavior Signature.** 2018. Masther's Degree Thesis in Computer Science, Federal University of Technology - Paraná. Ponta Grossa, 2018.

In a Grid, resources with low computational power, when directly competing with each other in a environment, could be undersused, favoring in time its scape. To reduce this effect, and allow a better usage of the environment resources, improving the grid potential, this study proposes a method of elements organization, who mades all of the most important resources caracteristics could be used to create the groups. To make it, the amount of each relevant resources component to the target application are compiled, composing his computational power. To allow an equal support demand capacity, the resources are arranged in groups, and their allocation is based on the computational power calculated. The separation of the groups is performed by a hierarchical algorithm based on the most distant element, due to the balance between strong and weak computational elements that it provides, thus generating groups with more similar characteristics. This organization allows that when there is a need to distribute a task, any group is able to execute it, becouse all have an equivalent computational power. Nevertheless, the groups are distinguished by the use that each of their resources has, which is not repeated perfectly in other groups. Each resource has a usage pattern, which over time can be refined till it could be predicted, when a group is generated it starts to display time windows that represent the best moments in which the resources can be used. These windows are represented by a behavior profile called Behavioral Signature. After determined the signature of each resources group, the signature of the entire group is also determined, which reflects the availability of its resources. As all groups have similar conditions to execute a demand, the signature allows their distinction, showing the most opportune moment to use each one, providing a metric to choose the request destination.

Keywords: Cluster. Behavior Signature. Load Balance. Hierarchical Algorithm. Grid.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACO	*Ant Colony Optimization*
CPU	*Central Processing Unit*
HD	*Hard Disk*
HMM	*Hidden Markov Model*
MA	*Moving Average*
PCA	*Principal Component Analysis*
RAM	*Random Access Memory*
ROC	*Rate of Change*
Th	*Threshold*
TS	*Timeslot*

SUMÁRIO

1 INTRODUÇÃO

É de conhecimento que diariamente novas tecnologias são dispostas no mercado, isto gera um efeito de descarte de equipamentos antigos, uma vez que em várias situações onde ao invés de se investir na melhoria do equipamento que possui se opta pela compra de um novo.

Situações como esta fazem com que a estimativa do número de equipamentos como computadores, celulares e TVs descartados tenha aumentado a partir de 2017, passando para 65.4 milhões de toneladas anualmente (COLLINS et al, 2013), representando um aumento de um terço em relação aos números de anos anteriores. Por isso, em um momento em que a comunidade global cobra pela manutenção de um meio-ambiente mais limpo, a criação de uma forma responsável de aproveitamento de equipamentos também passa pela necessidade de se melhor utilizar os que ainda funcionam.

Infraestruturas como Grade Computacional podem proporcionar a exploração de recursos mal utilizados, permitindo que uma aplicação existente em um máquina possa ser executada em uma outra (FERREIRA et al, 2003). Porém, para que estes recursos possam ser dispostos configurando uma Grade Computacional, ou *Grid*, é necessário que alguns pontos sejam atendidos (FOSTER, 2002):

- Os recursos coordenados não estão sujeitos a um controle centralizado: Em um *Grid*, é comum que se controle recursos que estão sob diferentes domínios, como por exemplo diferentes unidades administrativas da mesma empresa;
- Utilização de interfaces e protocolos de padrões abertos: Uma Grade Computacional é composta por elementos de várias plataformas, portanto para que possam sem comunicar é importante que o padrão de autenticação, autorização, descoberta de recursos e acesso a recursos seja em um padrão aberto;
- Entrega de um QoS *(Quality of Service)* diferenciado: Um *Grid* permite a entrega de recursos que coordenados garantem disponibilidade, tempo de resposta, e alocação de múltiplos recursos para atender demandas complexas, de modo todas estas características combinadas são maiores que a soma de suas partes.

Apesar de não ser uma tecnologia nova, os *Grids* raramente são utilizados como foram projetados para ser. Desenvolvedores não tem aproveitado seu poder para resolver uma única aplicação através de máquinas geograficamente distantes, mas sim envolvem a tecnologia para resolver problemas de aplicações com execuções em paralelo (SCHOPF, 2002). Em contra partida se não existem tecnologias que utilizem as grades computacionais, os desenvolvedores também acabam perdendo o foco de para quem desenvolver, quais as necessidades devem ser sanadas, por isso mais estudos e melhorias nesta área são necessários para se difundir mais sua utilização.

Grade Computacional leva em consideração um volume de recursos que pode variar até um nível global, por isso é importante que estes sejam dispostos em um formato que permita um gerenciamento mais simplificado. Neste trabalho, diferente de outras aplicações onde a organização dos recursos os coloca em concorrência direta uns aos outros, é feita a opção pela organização dos recursos em *clusters*, pois enquanto uma comunicação direta do recurso com o sistema cria uma dependência de gerenciamento unitária sobre ele, quando se está imerso em um grupo o número de recursos pode ser mais especializado, sendo que nele pode-se atribuir a um dos recursos a gerência sobre todos, diminuindo a complexidade geral em virtude da redução de elementos a se monitorar. Aliado a isso, em um *cluster* o recurso tem uma concorrência menor comparada a uma organização onde estes concorrem diretamente uns com os outros.

Existem várias formas para se agrupar elementos, e a escolha de qual método aplicar depende entre outras coisas, da quantidade de elementos alvo. Os métodos são baseados em identificar grupos ou elementos que se assemelham entre si e também diferem de outros, por isso é necessário que primeiramente se defina qual a característica que se deseja utilizar como métrica para definir se elementos são similares (ANDERBERG, 1973), como por exemplo a capacidade de processamento, ou a distância geográfica, sendo no estudo em questão a capacidade computacional do recurso, que é uma compilação de todos os valores dos seus componentes relevantes no processo, permitindo que o recurso como um todo seja avaliado, e não somente um de seus componentes.

Os recursos são organizados em grupos através de um algoritmo de clusterização denominado Algoritmo Hierárquico, o qual será responsável por dividir os recursos em grupos homogêneos, baseado no poder computacional calculado de cada recurso, isto permitirá que recursos com baixa capacidade computacional não sejam analisados diretamente e descartados em uma concorrência direta.

Além de organizar os elementos, é necessário que se conheça também o perfil do *cluster* gerado. Como cada recurso possui uma identificação própria, é necessário fundir todos os elementos em uma única classificação para que se estabeleça um perfil para o *cluster*, e assim se conheça por exemplo os momentos em que ele possui uma maior disponibilidade para uma execução. Este tipo de perfil pode ser gerado através do monitoramento do recurso computacional. Através deste monitoramento é possível se identificar o padrão da utilização do recurso. Um método que permite a geração deste tipo de perfil é o de Assinatura Digital (CARVALHO et al, 2014), ou Assinatura Comportamental como será tratado neste estudo, em virtude da nomenclatura anterior se referir a um termo conhecido na área de certificação digital, e também pelo nome escolhido representar melhor como o recurso se comporta em sua linha temporal.

Conhecendo o perfil dos recursos, é possível compilá-los em outra assinatura que representará o *cluster*. Esta assinatura permite, já que os *clusters* possuem poder computacional semelhante, determinar qual dentre eles tem a melhor capacidade de execução de uma tarefa.

Estes passos da arquitetura descritos anteriormente, assim como alternativas de algoritmos, e métodos de Assinaturas Comportamentais estão detalhados no decorrer desta disser-

tação.

1.1 OBJETIVO

O objetivo geral desta pesquisa é estabelecer uma arquitetura de utilização de recursos, que possibilite que um equipamento com baixo poder computacional tenha condições de contribuir com o ambiente, diminuindo assim a fuga de recursos por falta de uso.

1.1.1 Objetivos Específicos

Para composição do objetivo final da pesquisa é necessário se atingir alguns elementos intermediários, sendo eles:

- Calcular o poder computacional do recurso associado ao sistema e quantificá-lo, compilando suas características como CPU(*Computer Process Unit*), memória RAM(*Read Only Memory*) e memória total disponível em um único valor;
- Organizar os recursos em *clusters* de valor computacional equilibrado em relação aos outros;
- Traçar o perfil dos recursos e dos *clusters* gerados através de uma mecanismo de assinatura comportamental;
- Proporcionar o aumento na utilização dos recursos computacionais de menor capacidade.

1.2 JUSTIFICATIVA

Na maior parte das empresas, existe uma grande quantidade de recursos computacionais que são subutilizados, prova disso é que a maior parte dos computadores desktops tem apenas 5% do seu tempo livre utilizado (FERREIRA et al, 2003). A organização dos computadores dentro de um ambiente de grade computacional se dá pela topologia na qual ele se encontra, fazendo com que os recursos nem sempre se agrupem da melhor forma possível.

Outro fator é que as máquinas que não estão em uso nem sempre possuem os melhores recursos, em muitas situações são equipamentos que foram substituídos por novos. Imersos em uma grade, devido a este baixo poder computacional comparado com equipamentos mais novos, acabam sendo pouco utilizados, e com o tempo, sem uso, acabam deixando o ambiente. Por isso uma arquitetura onde todos tenham capacidade de participar, mesmo que proporcionalmente às

suas condições, faria que ao invés dos recursos com baixo poder computacional deixassem o ambiente, outros fossem agregados.

Alternativas para organização dos recursos podem ser encontradas em vários artigos que tem por finalidade agrupá-los, porém as aplicações utilizam como fator para este agrupamento apenas uma das características da máquina, como a CPU. Isto faz com que sejam gerados grupos com capacidades computacionais divergentes entre si. O resultado de uma organização neste formato é que a frequência em que o grupo com recursos mais fortes receberá tarefas será muito maior, visto que sua capacidade de processamento é maior em relação ao grupo fraco.

Não basta alterar a característica com que o agrupamento é feito, é necessário que se altere a maneira como o poder computacional da máquina é calculado, permitindo com que vários componentes do recurso sejam levados em consideração.

Ainda assim, se o agrupamento for feito baseado no poder computacional total da máquina, estaremos retornando no mesmo problema de geração de grupos antagônicos, por isso também se faz necessário uma aplicação diferenciada na geração dos *clusters*. Quando um recurso é inserido em um *cluster*, as suas características específicas que antes eram perceptíveis acabam por se confundir com as do grupo. Os recursos perdem sua identidade própria para assumir a do grupo.

Neste sentido pode-se observar a necessidade de agrupar os recursos, gerando grupos que possuam um poder computacional similar, e permitindo que todos sejam similares quanto a capacidade e disponibilidade de contribuição com o ambiente. Para isso, cada recurso deve passar por um processo de identificação para se definir qual sua real capacidade computacional, que também pode-se compreender como a capacidade de contribuição que ele pode oferecer, e então a sua alocação em *clusters* homogêneos e com uma identidade definida.

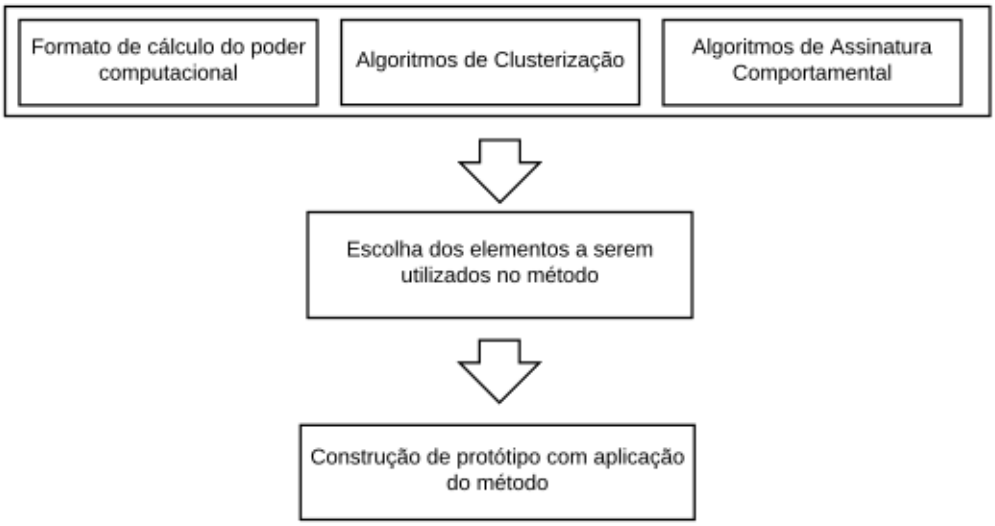
1.3 METODOLOGIA DE DESENVOLVIMENTO

Esta pesquisa foi elaborada conforme as etapas demonstradas na figura 1.

Os estudos envolvendo a fundamentação teórica foram divididos em três etapas, sendo elas: Formato de cálculo do poder computacional; Algoritmos de Clusterização; Algoritmos de Assinatura Comportamental. Estas etapas são executadas da seguinte maneira:

- Formato de cálculo do poder computacional: Durante esta etapa foram pesquisados formatos de estimativa de cálculo computacional de um recursos para que ao final fosse possível utilizar ou criar um método. Apesar de não ser encontrada dentro das pesquisas realizadas, a aplicação de um modelo de redes neurais perceptron se apresentou como uma alternativa, por isso modelos neste ambiente também foram elencados;
- Algoritmos de Clusterização: Para definição do algoritmo a ser utilizado para clusteri-

Figura 1 – Sequência aplicada no desenvolvimento do estudo.



Fonte: O Autor

zação foram pesquisados estudos que aplicam agrupamento de elementos. O algoritmo *k-means* está presente em grande parte das propostas de clusterização, e por isso houve uma preocupação na sua aplicação, considerando suas vantagens e desvantagens. O algoritmo hierárquico também foi alvo de pesquisa, o qual possibilita 3 tipos de abordagens, sendo elas por vizinho mais próximo, distância média entre vizinhos e vizinho mais distante. Também se pesquisou sobre processos de clusterização que envolviam duas etapas, como aplicação do *k-means* em um primeiro momento e após a aplicação de um algoritmo hierárquico para conclusão do processo;

- Algoritmos de Assinatura Comportamental: Para se definir como seria feita a identificação dos *clusters* e entender melhor o processo de elaboração de um perfil para eles, assim como suas vantagens, foi necessário se aprofundar em algoritmos como *Principal Component Analysis* (PCA), *Ant Colony Optimization* (ANT), *Holt-Winters* (CARVALHO et al, 2014) e *k-means*.

No final de cada etapa, algumas ideias foram adotadas para o desenvolvimento do método proposto. Para o cálculo do poder computacional não existe algo específico nas aplicações encontradas, então foi elaborado um modelo trazendo um conceito de redes neurais, porém aplicando valores das características dos recursos computacionais. Com os algoritmos foi necessário avaliar qual deles se comportava melhor em um ambiente com um grande número de elementos, e como era o resultado do seu processamento. Entre os algoritmos de clusterização havia a necessidade de que os *clusters* gerados fossem equivalentes, por isso algoritmos que geram grupos com diferenças acentuadas entre si foram descartados. Para a escolha do algoritmo de Assinatura Comportamental, foi levado em consideração qual deles poderia proporcionar uma visualização mais clara do perfil do *cluster*, baseando nos dados de cada recurso nele agrupado.

Visando demonstrar o funcionamento e avaliar os resultados foi desenvolvido um pro-

tótipo utilizando estes conceitos. Com isso foi possível coletar dados e avaliar o desempenho, assim como seus pontos fortes e suas limitações.

1.4 ORGANIZAÇÃO

Este trabalho está organizado da seguinte forma: No Capítulo 2 é descrito o funcionamento de uma grade computacional e sua abrangência, incluindo alguns modos disponíveis de clusterização e seus respectivos métodos. No Capítulo 3 é feito uma descrição do modelo de geração de uma assinatura de comportamento, qual sua técnica e a partir de quais estudos este modelo foi idealizado. No Capítulo 4 é demonstrado qual o modelo utilizado para compor o valor final do poder computacional dos recursos utilizando o conceito de redes neurais perceptron. No Capítulo 5 é descrita qual a abordagem escolhida para compor todo o método para a organização do estudo, envolvendo os três itens estudados anteriormente, grades computacionais com clusterização, assinatura comportamental e cálculo do poder computacional do recurso. No Capítulo 6 é demonstrada a construção do protótipo com a aplicação de toda a abordagem escolhida assim como os resultados obtidos pela sua implantação. No Capítulo 7 são expostas as conclusões oriundas do estudo assim como as propostas para trabalhos futuros.

2 GRADES COMPUTACIONAIS

Dentro da computação distribuída, um dos conceitos que possui grande relevância é o de Grades Computacionais ou *Grids*. Seu conceito se baseia em integrar em um único ambiente vários recursos heterogêneos dispersos geograficamente (NOROUZI; AKBARPOUR, 2013), os quais não estão sujeitos a um controle centralizado, porém, o real e específico objetivo que permeia o seu conceito é a coordenação de recursos, compartilhando e resolvendo problemas dinamicamente em organizações virtuais e multi-institucionais.

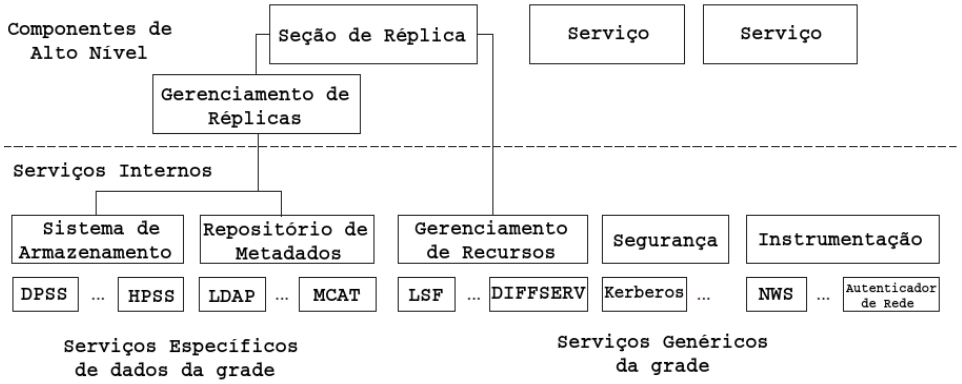
O compartilhamento de recursos não se refere a troca de arquivos, mas a se conectar diretamente a computadores, software, dados e outros recursos, e para que não hajam problemas, este compartilhamento deve ser altamente seguro (FOSTER; KESSELMAN; TUECKE, 2001). Ao se combinar os seus fatores as redes podem ter um efeito transformador no modo como a computação é executada e utilizada, não somente nos termos da computação, mas também no gerenciamento de dados (FOSTER, 1988).

As grades computacionais podem ser aplicadas de várias formas. O modo mais comum de aplicação de um *grid* consiste em rodar uma aplicação em outra máquina. A máquina onde esta aplicação roda normalmente pode estar com seus recursos utilizados em um dado momento devido a algum pico de atividade anormal. A aplicação em questão pode rodar em uma outra máquina que esteja em um estado de espera (FERREIRA et al, 2003). Aplicações em *grids* que fazem a utilização de máquinas com recursos pouco utilizados são chamadas de Grades Oportunistas.

São quatro os princípios que definem a arquitetura de um *grid*, e sua aplicação baseia-se no fato de que grades computacionais trabalham em um ambiente amplo, heterogêneo, multi-institucional e não podem assumir uma uniformidade ou política (FOSTER et al, 2000):

- Mecanismo de neutralidade: O acesso aos dados deve ser o mais transparente possível, eliminando a necessidade de se conhecer como os dados são gravados e resgatados, mantendo o sistema longe de detalhes de baixo nível;
- Política de neutralidade: A estruturação da grade é desenhada permitindo que decisões com implicâncias significativas de desempenho sejam expostas ao usuário ao invés de serem ocultadas internamente, assim, enquanto transmissão de dados e catalogo de réplicas são operações básicas, as políticas de replicação são levadas a alto nível, permitindo que sejam estipuladas em código de aplicação;
- Compatibilidade com infraestrutura da grade: Como o objetivo é atingir um campo amplo de recursos, os serviços básicos como autenticação, gerenciamento de recursos e informação foi estruturado, com isso mais ferramentas de baixo nível são compatíveis com mecanismos de *grids*;

Figura 2 – Principais componentes e estrutura da arquitetura de dados de um *Grid*



Fonte: Adaptado de (FOSTER et al, 2000)

- Uniformidade da estrutura de informação: Assim como na compatibilidade de infraestrutura, os dados também são estruturados, na prática isso significa que todos possuem o mesmo padrão de modelo de dados e interface de acesso aos metadados e catálogo de réplicas.

A implementação destes princípios proporciona a criação de uma arquitetura em camadas, na qual as camadas de nível mais baixo proporcional mecanismos de alta performance de acesso a uma relação de comandos básicos, enquanto as camadas de nível mais alto permitem a implementação de políticas da grade.

Na figura 2 percebe-se a separação relatada pelos princípios. Acima da linha pontilhada ficam os elementos de alto nível, como serviços, que permitem uma customização de acordo com a implantação alvo. Abaixo os elementos de baixo nível, que possuem um padrão de acesso aos elementos internos, tornando transparente para o utilizador das interfaces os detalhes de controle de elementos físicos por exemplo. Assim interfaces como o repositório de metadados pode implementar a comunicação com o gerenciador de diretórios LDAP, sem que o usuário necessite conhecer os detalhes internos da sua implementação.

Na literatura se encontram várias terminologias que classificam as grades, isso se deve ao fato de que sua aplicação muda de acordo com a necessidade de implementação. Portanto, para que se identifique melhor o formato de cada grade é necessário que seja feita uma classificação sobre a qual está implementada, podendo ela ser abordada segundo estes 3 tipos de topologias: de acordo com a aproximação dos recursos; de acordo com o nível de complexidade; e de acordo com a finalidade (MARIN; CAMARA; FUENTES, 2009).

As classificadas de acordo com a aproximação dos recursos, consideram o local aonde estão, fazendo distinção entre *grids* organizados em uma rede local com poucas máquinas e em um ambiente controlado, *grids* em uma intranet departamental ou corporativo e por último em um ambiente que abrange vários modelos de redes como *intranet]* e *internet*.

Quando classificada de acordo com o nível de complexidade, o que se é levado em conta é a localidade, quantidade e constituição dos recursos envolvidos. A localidade e a quanti-

dade fazem diferença ao se comparar um grupo em uma rede local com uma quantidade menor ou maior de recursos, mudando dessa forma sua classificação. A constituição diz respeito as características de cada recurso, como seu hardware ou o software. A classificação mais ampla contempla um número grande de máquinas, dispostas em várias localidades, em redes diferentes e com sistemas heterogêneos.

Por último, a classificação de acordo com a finalidade diferencia arquiteturas que proporcionam soluções para vários problemas de relacionamento entre empresas. Suas classificações pode ser considerada como computacional ao abranger sistemas voltados a exploração do poder de processamento dos recursos, *grids* de dados dedicados a tomar conta de um volume muito grande de informações, colaborativo, dedicado a proporcionar um ambiente em que recursos podem interagir mesmo estando geograficamente dispersos ou, *grid* utilitário, o qual proporciona um fornecimento de capacidades computacionais tal qual serviços públicos como de eletricidade, gerenciando picos de uso e incentivando usuários a alterarem seus hábitos através de incentivos econômicos.

Classificar corretamente o modelo de grade que está sendo utilizada ajuda a compreender qual o formato de aplicação se está sendo construída. Estes formatos ao invés de simplesmente separar em opções, também representam um crescimento de requisitos partindo de aplicações em um ambiente controlado e experimental para um de missão crítica (DZUBECK, 2002).

Dentro de um ambiente de grade computacional existem dois tipos de envolvidos, clientes e participantes. Clientes são indivíduos ou organizações com tarefas computacionais a serem executadas e dados a serem processados. Os participantes consistem também em indivíduos ou organizações, porém estes estão dispostos a aceitar uma porção dentre um total de tarefas para serem executadas em suas máquinas. O operador do *grid* é responsável por todas as suas execuções, incluindo o número de despesas relacionadas ao *link* entre os clientes e participantes (TAYLOR, 2006).

Uma característica em um ambiente *grid* é que recursos são compartilhados entre inúmeras aplicações, por isso a quantidade de recursos disponível para qualquer aplicação varia. Por isso, balanceamento de carga executa um importante papel, pois através do paralelismo e uma distribuição apropriada das tarefas é capaz de diminuir o tempo de resposta de uma aplicação (KAMARUNISHA; RANICHANDRA; RAJAGOPAL, 2011).

Neste contexto, clusterização é um dos mais importantes métodos para condições de balanceamento de carga, e uma das suas principais ideias é a de transparecer para o usuário externo que toda a relação de recursos é na verdade um único sistema (NOROUZI; AKBARPOUR, 2013).

2.1 CLUSTERIZAÇÃO

Agrupar elementos de acordo com características similares entre si não possui aplicação única na computação. Ela pode ser utilizada em diferentes situações como na área de *marketing*, para identificar comportamentos de um grupo de pessoas e assim destinar publicidades específicas àquelas que se enquadram no perfil, ou ainda na medicina onde um padrão de características pode ser identificado em pacientes com determinado tipo de doença. Porém quando utilizamos estes conceitos para aplicar a recursos computacionais, o termo *cluster* é mais apropriado.

Este processo de descoberta de conhecimento pode ser aplicado em bases com grande volume de dados, sendo que seu uso é comum na área de *data mining*. Implementações de serviços de data mining aliadas aos algoritmos de clusterização são uma das implementações de *grids* (YILDIRIM; UZUNOGLU, 2015).

O termo *cluster* não se refere a um método ou modelo particular, existem vários modos de separar os elementos em grupos e, a escolha do método depende entre outras coisas do número de elementos. Além disso, métodos normalmente utilizados para pequenos grupos costumam ser impraticáveis quando aplicados a um conjunto muito grande (NORUSIS, 2011).

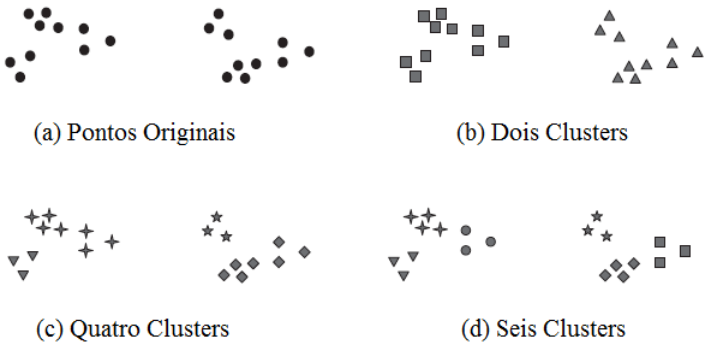
As soluções de clusterização por algoritmos, primeiro selecionam os nós que compõe a cabeça do *cluster*, e então outros nós que serão afiliados a ela, o que leva a diferentes *clusters*. Normalmente um peso é associado a cada nó, e o nó com maior peso é selecionado para ser a cabeça do *cluster* (MASSIN; MARTRET; CIBLAT, 2015). Este peso pode ser o identificador do nó como o MAC *address*, o seu poder computacional, quantidade de bateria restante, ou ainda uma união de vários deles.

Como existem vários tipos de algoritmos de clusterização, também existem vários objetivos para cada um, e com base neles os algoritmos podem ser classificados (MASSIN; MARTRET; CIBLAT, 2015). Contudo, a percepção da forma da aplicação do *cluster* em uma aplicação nem sempre é bem definida. A ideia se apresenta ainda mais difusa devido a deficiência de publicações na área. O problema se torna cíclico, ao passo que para que uma novidade seja publicada é necessária uma nova necessidade seja criada pelas aplicações, porém, se novas aplicações utilizando o conceito não são desenvolvidas não existem novas necessidades a se dar suporte, voltando ao início do processo (TAN; STEINBACH; KUMAR, 2005).

Na Figura 3 é possível perceber a diferença de aplicação e a dificuldade em se determinar o que utilizar na definição de um *cluster*. Neste exemplo, marcadores indicam a qual *cluster* cada elemento pertence. Para a mesma quantidade de elementos resultam quatro tipos diferentes de *clusters*, ainda assim o resultado final poderia ser diferente de acordo com o algoritmo abordado. Esta figura ilustra que a definição do *cluster* é imprecisa e, identificar qual o formato ideal a se aplicar depende da natureza dos dados e do resultado desejado, o que nem sempre é uma tarefa fácil.

Como a clusterização pode ser utilizada para agrupar elementos de acordo com um pa-

Figura 3 – Diferentes formas de clusterizar a mesma relação de pontos.



Fonte: (TAN; STEINBACH; KUMAR, 2005)

drão, ela também pode ser considerada como uma forma de classificação que cria um rótulo, atribuindo-o a um determinado grupo. Porém estes dados são provenientes apenas dos dados apresentados para o algoritmo de clusterização. Este formato é chamado de classificação supervisionada, pois os próprios rótulos atribuídos aos grupos assim como as características que os fazem semelhantes já são conhecidos anteriormente ao processo. Ao contrário disso, um processo não supervisionado é aquele que não atribui rótulo aos grupos, que é o caso do exemplo da Figura 3, onde os grupos são gerados com base em um padrão identificado no momento da clusterização (TAN; STEINBACH; KUMAR, 2005), o que permite que os elementos sejam agrupados de maneiras distintas, de acordo com as características identificadas pelo algoritmo.

Este exemplo é apenas uma das formas disponíveis de agrupamento, sendo ela um dos formatos mais conhecidos na literatura, aonde os elementos são agrupados de acordo com uma distância média a um ponto central. Porém várias abordagens são possíveis, pois como cada algoritmo resulta em um tipo de *cluster*, em alguns casos é necessária aplicação de mais de um deles em etapas distintas.

Em certas circunstâncias os *clusters* gerados não abrangem todos os elementos disponíveis na relação informada, isto acontece porque nem sempre é possível classificar todos os elementos. Quando um fica de fora das classificações, é porque suas características não são compatíveis com os padrões definidos para de obtenção dos grupos, isto não significa que o item é bom ou ruim, certo ou errado, apenas que para o tipo de algoritmo aplicado não é possível enquadrá-lo. Portanto, determinados algoritmos podem deixar de lado elementos que no resultado final podem ser significativos, enquanto outros algoritmos podem incluí-los. Estes são conhecidos pelo nome de *outliers*, ou ruídos, que são elementos que possuem uma natureza incerta, apresentam um comportamento inesperado ou então propriedades anormais (JIANG; SUI; CAO, 2010).

No processo de clusterização, inicia-se com uma relação de elementos os quais se deseja subdividir em grupos homogêneos. Para isso, primeiramente se escolhe quais variáveis

deverão ser levadas em consideração para se determinar com o que os grupos deverão ser similares. Tendo relacionado as variáveis, deve ser decidido se elas devem ser padronizadas com base em algum critério para que todas contribuam igualmente para a distância ou similaridade entre os grupos. Por fim se decide qual o método de clusterização deve ser aplicado, baseado no número de *clusters* desejados e nos tipos de variáveis que se deseja utilizar para formar os grupos (NORUSIS, 2011).

Após estas escolhas, para cada tipo de método possível de clusterização existem etapas que devem ser seguidas para obtenção do resultado final. Em certos casos se opta pela aplicação de mais de um método, e o resultado final é composto por um procedimento executado em duas etapas, os quais normalmente são aplicados em casos que possuem problemas muito grandes a serem tratados. Neste tipo de aplicação, primeiramente os dados passam por um dos algoritmos e resultam em chamados pré-clusters, os quais são a entrada para um segundo método que gera os *clusters* finais.

Para a escolha do método é necessário primeiramente conhecê-los, portanto na sequência é apresentada uma relação de algoritmos já conceituados que podem fazer a composição dos métodos disponíveis.

2.1.1 Algoritmo *k-means*

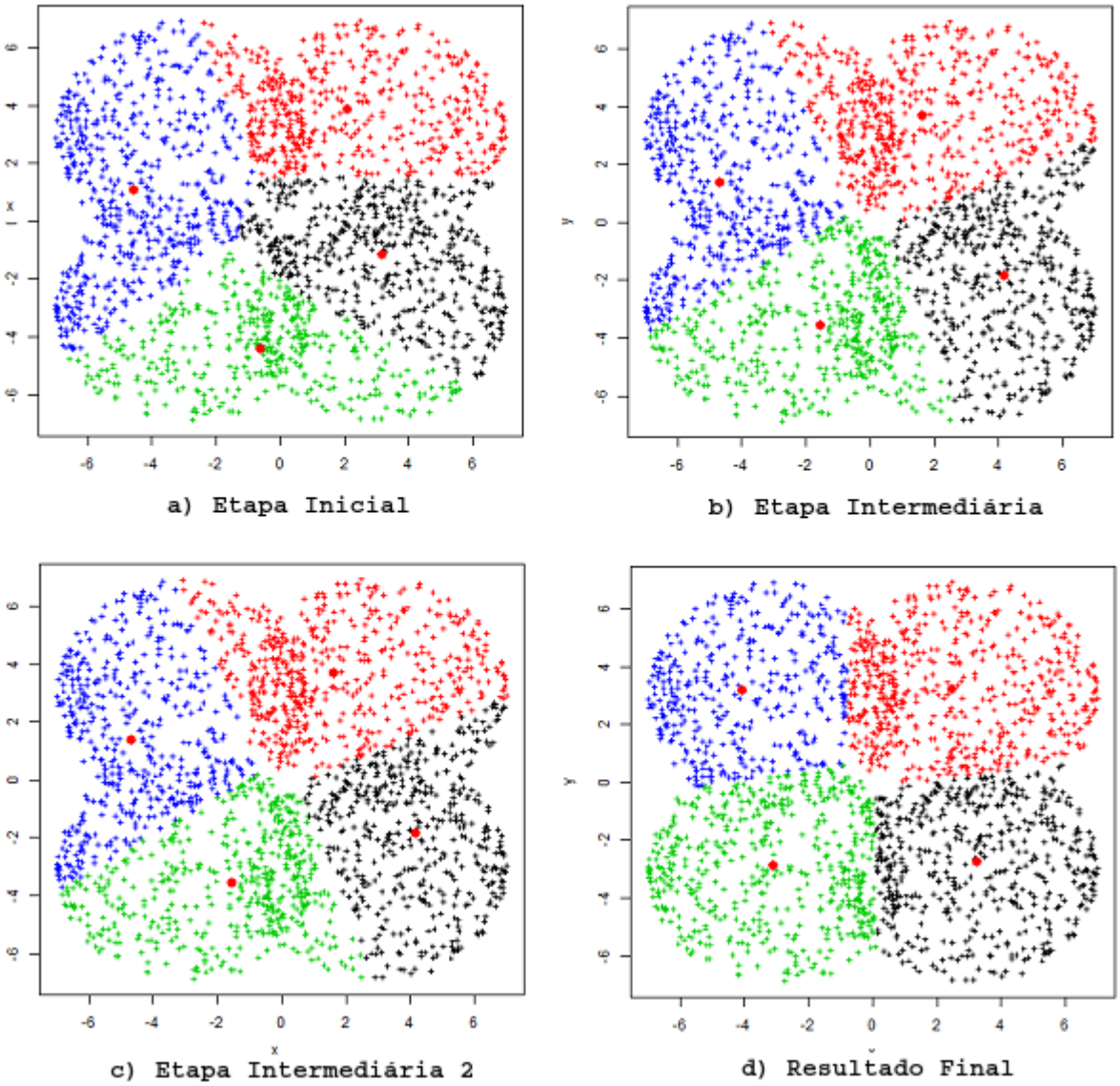
Em alguns modelos de algoritmos de clusterização o objetivo é criar clusters com objetos internos semelhantes, e comparados aos objetos de outros clusters, que estes sejam muito diferentes (HANMIN; HAO; QIANTING, 2012). O k-means, proposto em 1967 por J.B. MacQueen é um típico algoritmo baseado em distância, sendo hoje um dos mais populares algoritmos de clusterização (FORGY, 1965).

Seu nome é composto pela mistura de dois fatores do processo, "K"que indica o número de *clusters* desejados, e "means"que indica que os elementos devem estar unidos através de uma mesma distância média de um centroide. Seu conceito está norteado em que (BRADLEY; FAYYAD, 1998):

- Cada cluster é modelado por uma distribuição gaussiana esférica;
- Cada elemento é atribuído a apenas um cluster;
- Os pesos da mistura sejam iguais.

Seu funcionamento inicia com a escolha randômica de K elementos como K centroides e atribuindo cada elemento novo aos centroide mais próximo. Cada vez que um novo elemento é adicionado a um dos *clusters*, o seu centroide deve ser recalculado e ajustado. Este processo segue até que todos os elementos tenham sido atribuídos e nenhuma outra alteração nos centroides seja feita (HANMIN; HAO; QIANTING, 2012).

Figura 4 – Etapas do processo de clusterização com algoritmo k-means.



Fonte: Adaptado de (FARELLY, 2012)

Para atribuir o elemento ao centroide mais próximo é necessário uma medida que quantifique a noção de proximidade. Distância euclidiana frequentemente é utilizada em casos onde se estuda elementos em um ambiente cartográfico, enquanto similaridade por cosseno é mais indicada para documentos (NORUSIS, 2011). A distância euclidiana pode ser exemplificada como o espaço físico entre dois pontos, como calcular a distância entre Curitiba e São Paulo.

Na Figura 4 pode-se perceber o funcionamento das etapas do algoritmo. Dentro de todo o cenário de recursos presentes no relação dos dados, para cada recurso avaliado o centro é reposicionado. No cenário demonstrado na figura apenas duas etapas intermediárias estão presentes, representadas por (b) e (c). Na etapa inicial (a) os centros são escolhidos aleatoriamente e posicionados, e para cada etapa intermediária (cada novo elemento avaliado), assim como acontece em (b) e (c) os centros são reposicionados, se aproximando cada vez mais do resultado final para

o número de *clusters* solicitados. Ao término do processo todos os elementos foram avaliados e os centros dos *clusters* passam a não sofrer mais alterações, resultando na figura da etapa (d).

Como a escolha do número de grupos é feita por quem implementa a aplicação e não pelo próprio algoritmo, existe a possibilidade de não se obter o número ideal de *clusters*. Quanto maior o número de *clusters* solicitado mais específico será o grupo gerado, assim se a aplicação chegar a um extremo de solicitar um número tão próximo ao número de elementos, os *clusters* terão ao final somente um elemento, tornando o agrupamento desnecessário. Mesmo sendo aplicado em muitos estudos, não significa que ele sempre converge na partição ideal dos elementos, porém existem casos em que ele o fará (MACQUEEN, 1967).

Apesar dos benefícios relacionados a fase de implementação, foi verificado em experimentos que o algoritmo não é tão bom em uma aplicação real em larga escala (FANG; JIN; MA, 1967).

2.1.2 Algoritmo Hierárquico

De uma forma mais abrangente, algoritmos podem ser classificados de duas formas, hierárquicos e não hierárquicos. A diferença mais significante para os algoritmos não hierárquicos é o fato de que o número de *clusters* não precisa ser informado (DAVIDSON; RAVI, 2005) para clusterização hierárquica. Posto isso, o algoritmo *K-means* citado anteriormente se classifica como não hierárquico, visto que é necessário informar o número de *clusters* desejado antes do processamento.

É de conhecimento que melhores resultados podem ser obtidos de clusterizações hierárquicas para muitos conjuntos de dados no caso de fronteiras não lineares entre *clusters*. Existem dois métodos para sua aplicação, um denominado divisivo e outro chamado aglomerativo (AHC - *Agglomeratic Hierarchical Clustering*), o qual é geralmente utilizado (ENDO; HAMASUNA; MIYAMOTO, 2007). A diferença entre ambos pode ser percebida mais facilmente analisando as etapas do funcionamento das duas modalidades. Pode-se descrever o modo aglomerativo como (TAN; STEINBACH; KUMAR, 2005):

- Considera cada elemento como um cluster;
- Calcula o nível de similaridade entre todos os pares de *cluster* e combina os pares de acordo com o nível estabelecido;
- Recalcula o nível entre o novo *cluster* e outro.

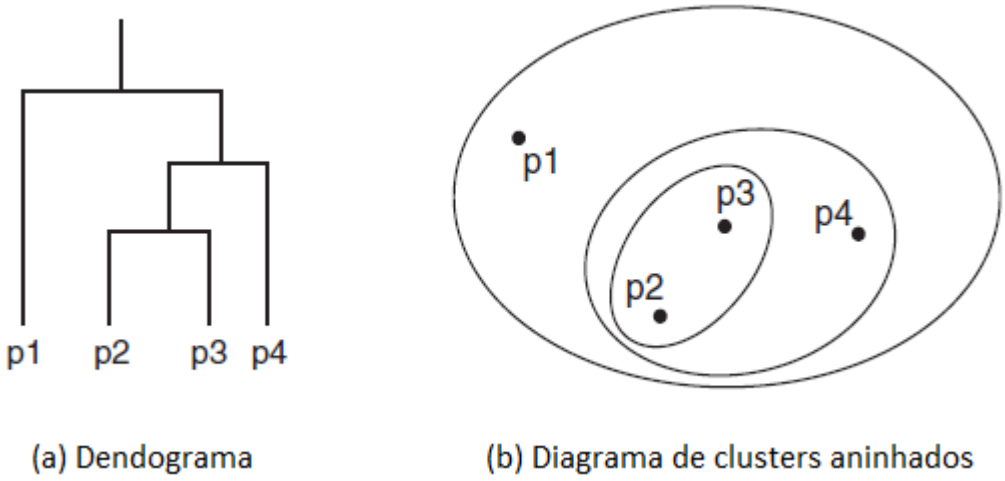
O processo divisivo inicia ao contrário do aglomerativo, sendo que ele (TAN; STEINBACH; KUMAR, 2005):

- Inicia considerando que toda a relação de elementos é um único *cluster*;

- A cada etapa divide o *cluster* segundo um padrão definido;
- Finaliza o processo quando todos os elementos estão divididos em *clusters* de elemento único.

Com isso, algoritmos hierárquicos como o próprio nome sugere geram uma relação de sub *clusters*, e são representados por um formado de gráfico no estilo árvore chamado dendogramas. Existe também a possibilidade de representá-lo no formado de grupos aninhados. Ambos os formatos podem ser visualizados na Figura 5.

Figura 5 – Clusterização hierárquica de quatro pontos demonstrada por um dendograma e *clusters* aninhados.



Fonte: (TAN; STEINBACH; KUMAR, 2005)

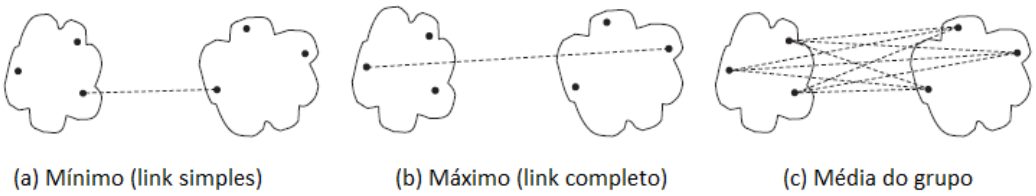
Este tipo de algoritmo não somente permite a um usuário escolher uma granularidade particular do *cluster*, mas em muitos domínios eles formam uma hierarquia, os quais são partes um dos outros, tal qual a fisiologia evolucionária de uma árvore.

O algoritmo aglomerativo é fácil de implementar e na sua execução segue reduzindo o número de *clusters* até atingir 1. Entretanto estes benefícios esbarram em questões de eficiência quando o número de dados é muito grande, onde as distâncias precisam ser recalculadas a cada nível do dendograma (DAVIDSON; RAVI, 2008).

Dentro da aplicação do algoritmo hierárquico existem três abordagens diferentes que implicam na forma como as etapas da clusterização atuam. Quando se faz a junção dos *clusters*, estes podem ser unidos com base na distância entre eles, podendo ser eles:

- Agrupamento por vizinho mais próximo;
- Agrupamento por vizinho mais distante;
- Agrupamento por distância média;

Figura 6 – Modos de clusterização do algoritmo hierárquico baseando-se na distância.



Fonte: (TAN; STEINBACH; KUMAR, 2005)

Esta diferença pode ser melhor visualizada na figura 6. O exemplo da figura (a) mostra como seria um agrupamento por vizinho mais distante, e neste caso, os subgrupos tendem a possuir características distintas comparados entre eles, mas internamente os subgrupos possuem elementos com características semelhantes (estão situados mais proximos uns aos outros), se comparado com o exemplo da Figura 5 o agrupamento primeiramente aconteceriam com p1 e p2 formando um subgrupo, e p3 e p4 em outro. No exemplo do dendograma da figura 6.b é demonstrado o agrupamento por vizinho mais distante, o qual em seu resultado final gera grupos com características mais equilibradas, e no caso do dendograma da Figura 5 seriam formados o subgrupo p1 e p4, e em outro subgrupo p2 e p3. Na situação da figura 6.c os elementos são agrupados com base na distância média entre todos os elementos, sua composição é equilibrada mas a junção promove um resultado diferenciado, onde ainda tomando como exemplo o dendograma da Figura 5 os subgrupos formariam no primeiro os elementos p1 e p3, e em um segundo subgrupo p2 e p4.

2.1.3 Algoritmo *Ant Colony Optimization*

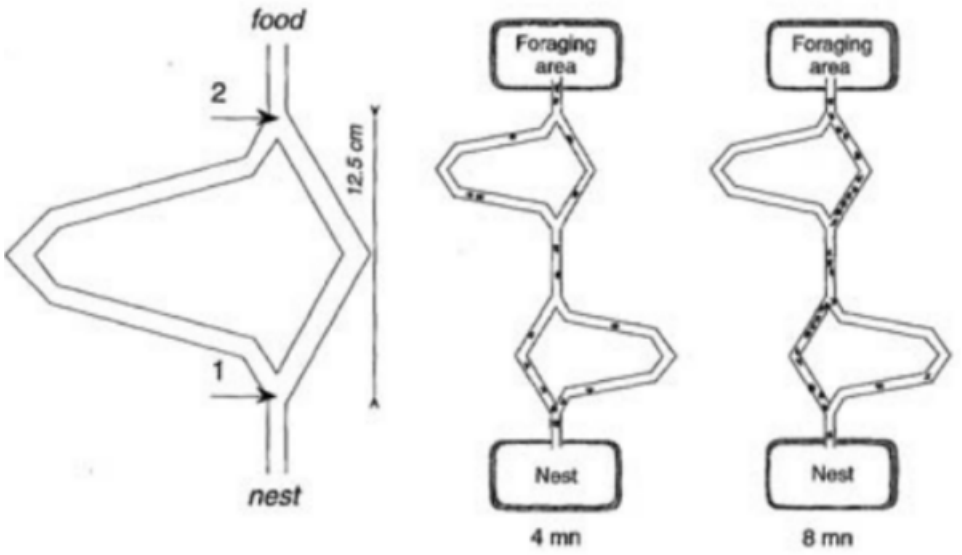
O algoritmo ACO(*Ant Colony Optimization*) é um algoritmo probabilístico, e é baseado no princípio de inteligência de enxame, termo que é utilizado para se designar sistemas aonde um comportamento coletivo de uma população causa a solução de problemas. O termo enxame apesar de denotar especificamente o coletivo de abelhas, no caso do algoritmo ele faz referência a qualquer coleção de elementos capazes de interagir. Neste formato de aplicação, agentes competem e cooperam globalmente em sincronia para encontrar uma solução mais aprimorada (DORIGO; BIRATTARI; STUTZLE, 2007).

O desenvolvimento do algoritmo foi baseado no comportamento das formigas, que fazem uso de feromônios para traçar o caminho. Quando precisam escolher um caminho elas tendem a escolher aquele que possui uma marcação mais forte de feromônio (BLUM, 2005), com isso os caminhos mais utilizados acabam por ter um traço mais forte, e aqueles que foram pouco ou deixaram de ser utilizados, aos poucos vão tendo seu traço de feromônio evaporado.

Isto pode ser percebido no estudo (BONABEAU; DORIGO; THERAULAZ, 1999),

aonde foi criado para um grupo de formigas um caminho longo e um curto entre o ninho e comida. O que aconteceu é que as formigas aos poucos foram ocupando ambos os caminhos, porém ao encontrarem a comida, aquelas formigas voltavam pelo mesmo caminho e deixavam novamente o traço de feromônio, tornando ele mais forte. Em alguns minutos o caminho mais longo deixava de ser ocupado e o mais curto passava a ser a forma escolhida.

Figura 7 – Definição do caminho por colônia de formigas.



Fonte: (BONABEAU; DORIGO; THERAULAZ, 1999)

Pode-se perceber na figura 7 o que foi executado no estudo. O resultado variava de acordo com a espécie das formigas, mas nesta em questão após oito minutos o caminho longo ja era praticamente inutilizado. A algumas espécies mesmo sendo apresentado apenas o caminho mais longo e após 30 minutos então ser apresentado o caminho mais curto, estas foram capazes de alterar a escolha para o caminho mais curto, isso mostra uma capacidade de adaptação em face a mudanças do ambiente.

As duas principais características que o diferenciam das outras formas de comunicação são (DORIGO; BIRATTARI; STUTZLE, 2007):

- É uma forma indireta de comunicação através do ambiente: insetos alteram informações modificando seu ambiente;
- A informação é local: ela pode ser acessada somente por aqueles insetos que visitam o local no qual ela foi armazenada (ou na sua vizinhança imediata).

Comparado a um sistema computacional, uma formiga poderia se movimentar randomicamente sobre um planisfério bidimensional produzindo informação aleatória, assim como um sistema poderia receber informações provenientes de qualquer situação neste ambiente. Desta forma um agrupamento de formigas é essencialmente parecido com um agrupamento de dados (JIANG; YU; GONG, 2010) sendo elas agentes dentro do processo.

Este formato pode ser utilizado para localização de caminhos como também pode ser aplicado para clusterização. Sua aplicação para clusterização ocorre em duas etapas: Fase de inicialização e fase de sorteio (HANDL; KNOWLES; DORIGO, 2003).

A fase de inicialização acontece em três etapas:

- Todos os itens de dados são aleatoriamente espalhados pelo *grid*;
- Cada agente escolhe aleatoriamente um item de dados;
- Cada agente é disposto em uma posição aleatória no *grid*.

Após o término da fase de inicialização acontece a fase de sorteio que ocorre no formato de um *looping* simples, e também é dividida em três partes:

- Um agente é selecionado aleatoriamente;
- Um agente da um passo de um tamanho predeterminado em uma direção aleatória no *grid*;
- O agente probabilisticamente decide se deixa o item de dado na posição.

Quando o agente se move, ele pode decidir se deixa o item de dado que possui na sua posição atual, desde que ela não esteja ocupada por outro item de dado e, caso esteja ocupada, uma posição vizinha é escolhida aleatoriamente. Durante a etapa de movimentação, o traço de feromônio é alterado em virtude da mudança de posicionamento dos item de dados dos agentes. Nesta situação, a quantidade de feromônio em cada turno é determinada por dois fatores (LI et al, 2016): O feromônio que evapora e o que é adicionado recentemente, ou seja, o elemento de dado que é movido e o elemento que é deixado pelo agente no novo local.

O resultado após as iterações pode ser melhor percebido na figura 8, onde é exibido uma simulação de clusterização, e nela, 4 círculos que mostram momentos subsequentes do processo de um *grid* de 200 de diâmetro, com 5000 elementos dispostos aleatoriamente no *grid*.

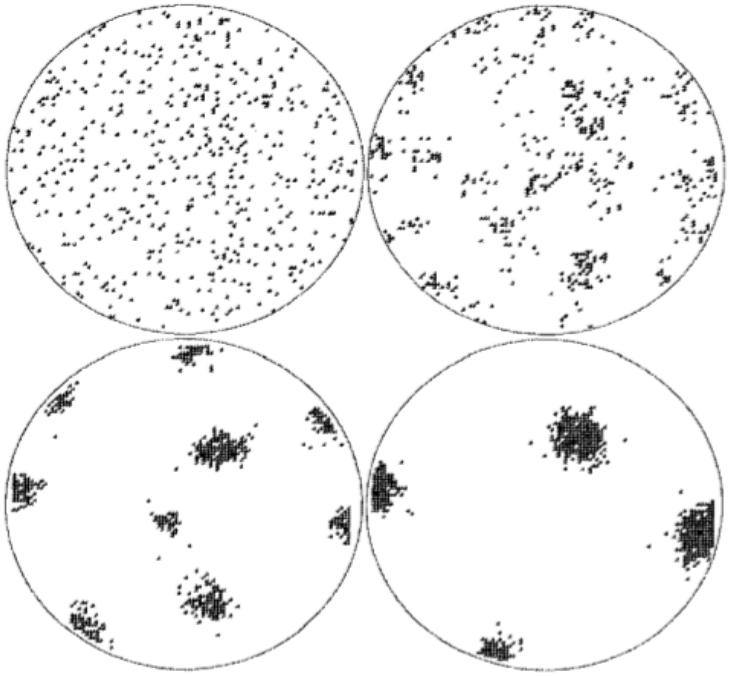
Tabela 1 – Comparação de tempo de execução entre ACO e K-means

	Tempo de Execução		
Número de elementos	699	1000	2500
ACO	10,54	15,68	19,2
k-means	0,06	0,74	46,04

Fonte: Adaptado de (HANDL; KNOWLES; DORIGO, 2003)

Em comparação com o algoritmo *k-means* citado anteriormente que também possui a funcionalidade de gerar *clusters* com um resultado final muito aproximado do algoritmo em questão, apesar do ACO possuir uma resposta ligeiramente mais lenta em se tratando de grupos com um número pequeno de elementos, quando o número aumenta sua resposta passa a ser mais

Figura 8 – Simulação de um modelo de clusterização.



Fonte: (BONABEAU; DORIGO; THERAULAZ, 1999)

eficiente, como pode ser visto na tabela 1 que reflete um estudo efetuado em (HANDL; KNOWLES; DORIGO, 2003), o qual atribui esta grande variação em virtude da escalabilidade dos algoritmos, onde ACO é linear e por isso é menos afetado pela alteração de dimensionabilidade.

Além disso o ACO pode trabalhar com qualquer tipo de dado e ainda possibilita que o sistema encontre automaticamente o número ideal de *clusters*, o que não ocorre no algoritmo *k-means*, onde é necessário que seja informado qual o número desejado.

A título de comparação com os outros algoritmos já expostos, o Quadro 2.1 organiza o objetivo, aplicação, vantagem e desvantagem destes. Nenhum deles tem como objetivo principal a criação de grupos balanceados, todos buscam estas uniões com base em métricas que de alguma forma refletem uma comparação entre os elementos a serem clusterizados. Contudo, na sua aplicação, o Algoritmo Hierarquico, apesar de não objetivar a criação de grupos balanceados, pela sua técnica de junção ao final, como exposto na tabela, tende a produzir estes grupos.

Cada algoritmo pode ter sua aplicação mais indicada em certas circunstâncias, isso não demonstra que um é melhor que outro, apenas mais indicado em determinadas situações. Os algoritmos K-means e ACO, em suas características agrupam os elementos que tem uma proximidade em relação a alguma métrica criada, no caso do k-means a centróide e no ACO a quantidade de feromônios no local. Por isso quando estes processos são executados o resultado do *cluster* não é uma preocupação, afinal é uma consequência do seu método. O Algoritmo Hierarquico da mesma forma não tem uma preocupação com o resultado, porém o seu método de junção dos elementos mesmo não sendo o objetivo, gera grupos mais balanceados, fator que nesta aplicação é essencial para o objetivo da arquitetura. Sendo assim, para o estudo aplicado, em virtude desta

Quadro 2.1 – Comparação de algoritmos de clusterização

	K-means	**Algoritmo Hierárquico**			**ACO**
		Vizinho mais Próximo	**Distância Média**	**Vizinho mais Distante**	
Objetivo	Agrupa elementos semelhantes	Prioriza elementos de valores mais próximos	Une os elementos com a menor média de distância entre eles	Une elementos com as características mais diferentes	Une elementos por aproximação
Aplicação	Identificar indivíduos com as mesmas características	Agrupar elementos por ordem de proximidade. Produz um grupo que se incrementa dos outros elementos próximos.	Agrupar elementos por ordem de distância média. Produz grupos de tamanhos variados.	Agrupar elementos distantes. Tende a produzir grupos mais balanceados	Agrupar elementos com base na proximidade
Vantagem	Tem um bom desempenho mesmo para volumes grandes de dados	Não necessita especificar o número de *clusters*	Não necessita especificar o número de *clusters*	Não necessita especificar o número de *clusters*	Pode trabalhar com qualquer tipo de dado; Não necessita especificar o número de *clusters*
Desvantagem	Necessário especificar o número de *clusters* desejado	Baixa eficiência com grande volume de Dados	Baixa eficiência com grande volume de Dados	Baixa eficiência com grande volume de Dados	Ligeiramente mais lento que algoritmos do mesmo tipo, como *k-means*

Fonte: O Autor

sua característica, o Algoritmo Hierarquico conquista seu lugar como mais indicado, sendo que ao tratar da metodologia mais a frente sua escolha será justificada na arquitetura.

3 ASSINATURA COMPORTAMENTAL

Também referida como Assinatura Digital, o termo remonta ao processo de disponibilização de uma determinada característica de um equipamento computacional em um formato visual de fácil compreensão. Este processo já foi aplicado para levantamento de um perfil de tráfego de uma rede (CARVALHO et al, 2014) e através dele é possível monitorar a situação de um equipamento, porém seu conceito permite uma aplicação mais ampla no âmbito da computação.

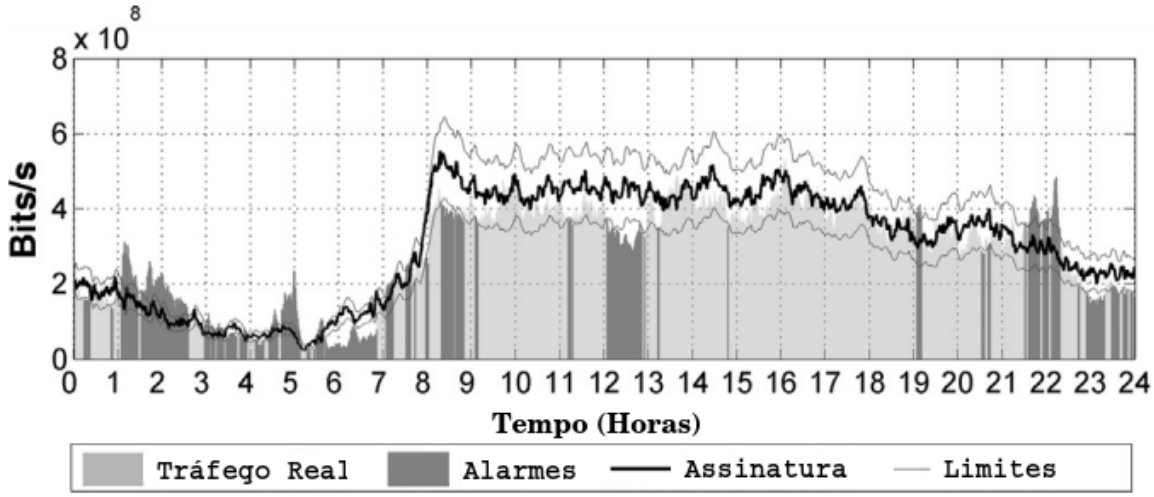
Em ambientes de alto desempenho é necessário que se tenha controle dos elementos computacionais disponíveis, desde o fluxo de dados que transmitem quanto a carga de trabalho que lhes é imposta, para isto é importante não somente se conhecer o dispositivo mas também o seu comportamento durante um período de funcionamento. A partir disso é possível detectar se um elemento está tendo um funcionamento inesperado ou ainda se seu poder computacional está sendo mal aproveitado.

Isto influencia muito na área de gerenciamento de redes, onde é importante que os equipamento estejam sempre trabalhando na sua melhor capacidade, assim em ambientes onde falhas podem ocorrer e resultar em grandes perdas estas podem ser minimizadas através de uma ação proativa, ou então de uma rápida resposta a um ponto de falha.

Para se determinar o comportamento de um recurso computacional e assim com uma maior segurança poder determinar como é o seu funcionamento regular dentro de um ambiente, é necessário se avaliar por um determinado período como ele se comporta regularmente.

Algoritmos anteriormente citados como *Ant Colony Optimization* e *k-means* já tiveram seus funcionamentos avaliados em alguns estudos como os executados em (DORIGO; BIRATTARI; STUTZLE, 2007), (HANDL; KNOWLES; DORIGO, 2003) e (OHNO et al, 2009) aplicando-os em conjunto com outras técnicas e comparando resultados.

Gráfico 1 – Assinatura Comportamental de Segmento de Rede Utilizando Análise de Fluxo.



Fonte: Adaptado de (CARVALHO et al, 2014)

No gráfico 1 é possível compreender melhor a representação da Assinatura Comportamental, que no estudo é citada como Assinatura Digital. Na figura é demonstrada a Assinatura gerada, assim como tráfego real dos dispositivo e as distorções detectadas em forma de alarme. Algo a se ressaltar é que a assinatura não está perfeitamente sobre o tráfego, isso se deve ao fato de que ela é criada e refinada de acordo com o tempo e composta pelo tráfego de vários dias analisados, portanto o que está representado na figura é o tráfego de um dado período do recurso, juntamente com a sua assinatura, permitindo assim analisar e validar os alarmes.

São poucos os estudos que apresentam uma definição de Assinatura Comportamental, por isso também não é grande o número de algoritmos aplicados a este tipo de aplicação. Existem formas diferentes de detecção de anomalias, como análise massiva de logs (LIU et al, 2018), ou ainda a utilização de sinais eletromagnéticos gerados involuntariamente por disposivitos para construir um padrão (SEHATBAKHSH et al, 2018), porém a Assinatura em si não é aplicada nestes estudos.

Alguns algoritmos são aplicados para geração desta assinatura, construindo também para cada um deles um resultado apesar de aproximado, diferente em detalhes, que variando a aplicação pode ser a diferença entre permitir aplicabilidade ou não. Em sequência cita-se alguns destes algoritmos aplicados em alguns estudos, seus resultados e como estes foram inseridos no contexto do estudo.

3.1 ASSINATURA COM ACO

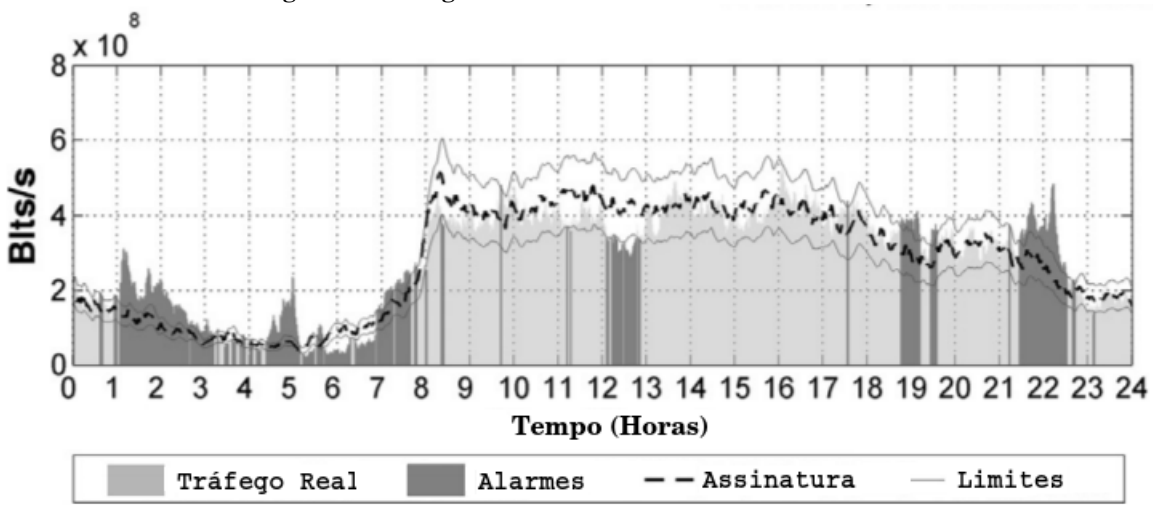
A Assinatura Comportamental pode ser comparada com um caminho traçado pelo uso do recurso ao longo do tempo, se observada a figura 1, horizontalmente a linha da assinatura traça por ”onde o recurso passou” durante o seu período de utilização. Esta percepção permite a aplicação do algoritmo ACO, já que ele auxilia na identificação de qual o caminho mais utilizado em questão por um determinado elemento.

Em (PROENçA et al, 2015) o algoritmo é utilizado para clusterizar os dados e identificar suas similaridades, já que através dele é possível a identificação de padrões, comportamentos e características.

Através da sua funcionalidade de simular o traço de feromônio das formigas, a Assinatura Comportamental é gerada pela repetição de valores aproximados, fortalecendo o traço do uso na região e enfraquecendo em locais com valores que ocorrem esporadicamente, determinando por fim qual é o traço natural de uso do recurso.

No gráfico 2 é demonstrada uma assinatura gerada com o algoritmo ACO no estudo de (CARVALHO et al, 2014), nela é possível visualizar assinatura com um sinal tracejado e o tráfego real. Esta assinatura é feita com base na análise de dados de vários dias, por isso o tráfego não encaixa perfeitamente na assinatura, pois existem alterações diárias que podem

Gráfico 2 – Assinatura gerada com algoritmo ACO.



Fonte: Adaptado de (CARVALHO et al, 2014)

refletir numa ocorrência distinta do natural, por isso o sistema que faz o monitoramento deve possuir um mecanismo para tratar este tipo de ocorrência.

3.2 ASSINATURA COM *K-MEANS*

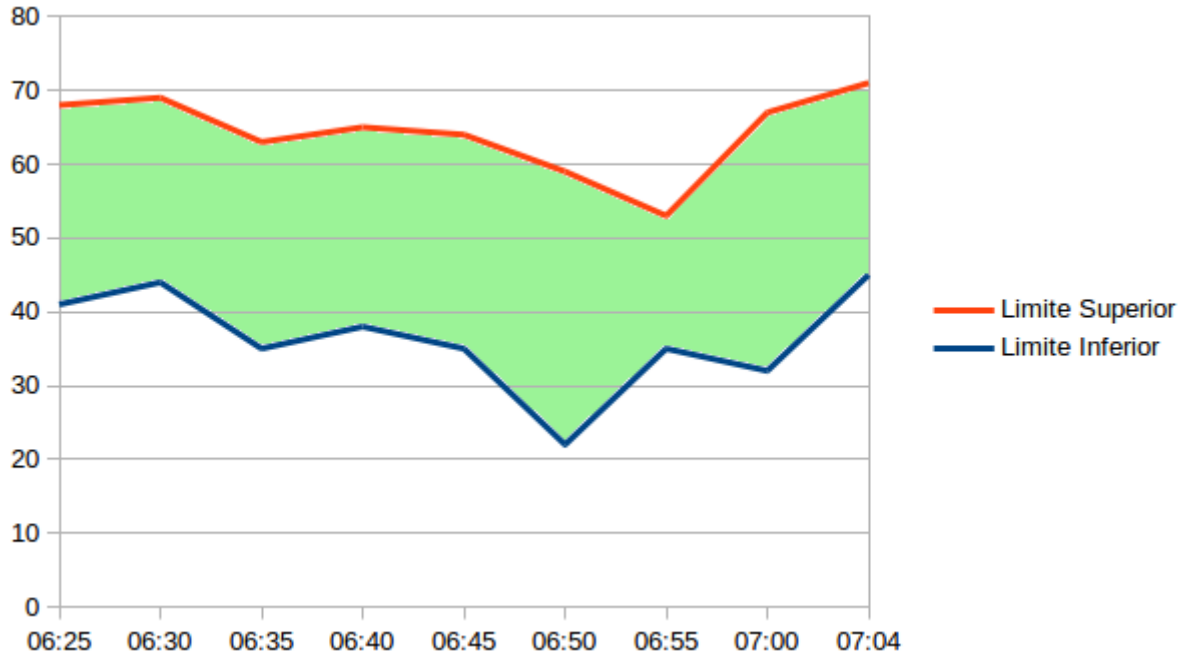
Como já descrito anteriormente, este algoritmo não foi criado propriamente para aplicações de padrões de caminhos ou sequências, mas sim para agrupar os elementos com características mais próximas.

Algumas abordagens utilizando este algoritmo são propostas e aplicadas em estudos como em (OHNO et al, 2009), que utiliza o *k-means* em conjunto com uma abordagem denominada HMM(*Hidden Markov Model*), que usa inferências estatísticas de probabilidade (BAUM; PETRIE, 1966), nele, é feita a descoberta do padrão submetendo os dados primeiramente ao algoritmo de clusterização, e em sequência seu resultado é dado como entrada para o método matemático. O estudo obteve bons resultados, com um baixo adicional de *overhead* ao ser adicionado no kernel do linux para funcionamento, e também um número mínimo de falsos negativos e positivos quando comparado a outros métodos como *Threshold*(Th), Moving Average(MA) and Rate of Change(ROC);

Em uma proposta de utilização com o algoritmo, em (SENGER; GOIS, 2017b) a clusterização é utilizada criando uma área de atuação, diferentemente de outras abordagens que costumam aplicar uma tolerância. Neste o algoritmo através do grupo formado de registros no mesmo instante de um determinado dia é que define quais os limites de atuação do recurso computacional em questão.

Esta abordagem pode ser visualizada e melhor compreendida no gráfico 3, onde a área em verde representa os limites em que o recurso funciona em um período normal, e uma ocor-

Gráfico 3 – Assinatura Comportamental e margem de uso de recurso computacional.



Fonte: Adaptado de (SENGER; GOIS, 2017b)

rência fora desta área denota uma anomalia na execução.

Apesar do estudo ainda não possuir resultados da aplicação, ele consegue exemplificar que o algoritmo *k-means*, que é aplicado para agrupamento de informações de características próximas, comumente aplicado em mineração de dados, também pode ter seu funcionamento agregado a uma abordagem de Assinatura Comportamental.

3.3 ALGORITMO PCA

Principal Component Analysis é um procedimento estatístico para problemas multivariados reduzir a dimensionalidade através da análise da variação de cada variável em conjunto com todas as dimensões. O método identifica os padrões de dados e pode reduzir o número de dimensões sem muita perda de informação, que pode então ser representado por uma relação de dimensões reduzidas (JOLLIFFE, 2002).

Sua aplicação se justifica por ser notoriamente sensível a *outliers* que se encontram em grande parte dos dados reais, e em muitos casos acabam por desviar a solução correta, justamente pela presença de uma pequena porção de pontos de dados periféricos, que não estão em conformidade com o modelo (HAUBERG et al, 2016).

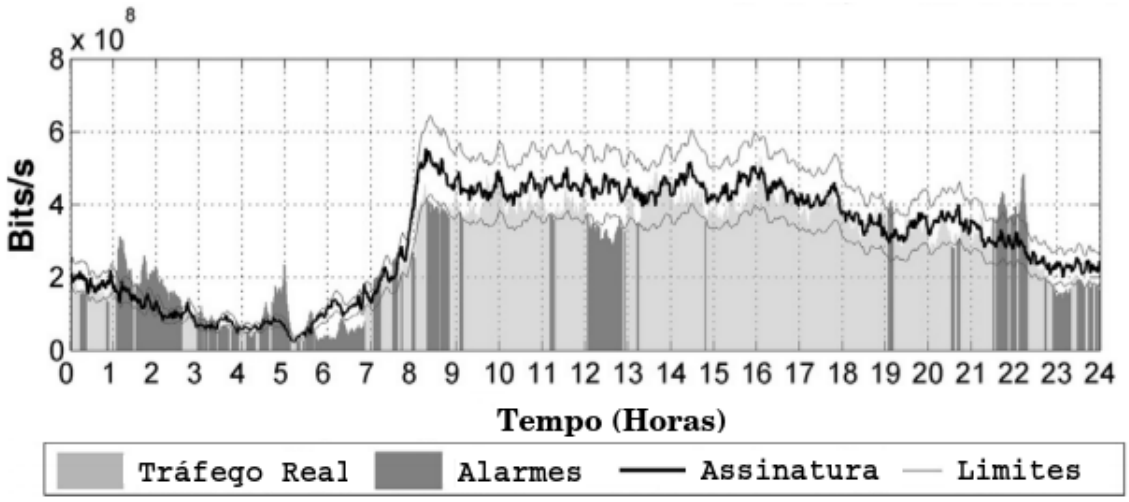
O funcionamento clássico deste algoritmo dispõe os dados em forma de uma matriz n x p, onde p são as colunas e representam as dimensões e, as linhas n são os n fluxo de dados coletados (PROENçA et al, 2015).

Para cada dimensão são calculados dois elementos importantes para a composição,

denominados auto-vetores e auto-valores, gerados através da covariância da matriz. Os valores utilizados na composição de uma Assinatura Comportamental envolvem apenas os auto-vetores.

Os auto-vetores utilizados são aqueles que possuem os auto-valores médios, pois, os maiores auto-valores significam a dimensão com uma maior variação entre todos os componentes, e utilizá-los para gerar a Assinatura Comportamental entraria em choque com o padrão de comportamento.

Gráfico 4 – Assinatura Comportamental gerada por PCA.



Fonte: Adaptado de (CARVALHO et al, 2014)

No gráfico 4 é demonstrada uma assinatura gerada com PCA. Os dados aplicados são os mesmos utilizados na figura 2, portanto é possível visualizar a diferença no resultado da aplicação de ambos os métodos.

4 PROPOSTA DE DISSERTAÇÃO

Ambientes de grades computacionais conectam recursos locais e remotos e oferecem seus momentos de baixo ciclos de processamento para serem utilizados em trabalhos que exigem um processamento mais intenso (HAIDER; NAZIR, 2017), assim, são provedores de recursos computacionais para outros que têm uma necessidade de consumo de recursos acima do que ele já possui. Visto pelo lado do consumidor, este ambiente precisa apenas que no momento em que sejam solicitados estes recursos, que estejam prontamente disponíveis para uso, porém do lado de quem disponibiliza o recurso nem sempre os elementos integrantes possuem a mesma forma de tratamento.

Em ambientes onde os recursos estão sob uma arquitetura responsável apenas por disponibilizar seu poder computacional, a não utilização de certos equipamentos ou mesmo a alta utilização de outros, nem sempre representa um problema para a estrutura, visto que, sob uma ótica de gerenciamento dos recursos como um todo, a soma dos seus poderes computacionais representam a quantidade disponível para fornecer o serviço, e se 70% da sua capacidade esteja sendo utilizada, não há a preocupação se estes 30% restantes estão distribuídos entre as máquinas ou então se são sempre os mesmos recursos que atendem a demanda e os restantes são acionados apenas quando a utilização passa da capacidade destes recursos iniciais. Por isso em casos de um serviço empresarial de disponibilização dos recursos computacionais, a não utilização de alguns dos recursos atrelados ao *grid* não corresponde a um problema, visto que apenas existe a necessidade da pronta disponibilidade da capacidade do recurso provedor no momento da requisição.

Porém existe ainda uma abordagem diferente da citada acima, como a construída em (GóIS, 2009), onde é proposta uma arquitetura para comercialização de serviços de *grids*, não utilizando o conceito de uma empresa provedora de serviços, mas sim como de um livre mercado, aonde pessoas podem utilizar os seus equipamentos e disponibilizá-los para serem provedores de recursos computacionais. Esta abordagem é interessante visto que ajuda a solucionar o problema dos equipamentos que entram em desuso tanto em residências quanto em empresas devido a aquisição de modelos mais avançados.

O funcionamento é próximo de um banco, onde clientes deixam seu dinheiro guardado e recebem por exemplo uma taxa x mensal de juros por isso, em contra partida o banco faz um empréstimo a outros clientes que por sua vez pagam uma taxa de $2x$ ao banco, e assim todos recebem algum reembolso. Mas isto funciona apenas por que todas as partes são de alguma forma beneficiadas, quem deixa o dinheiro no banco recebe por isso, o banco recebe também pelo serviço de empréstimo que presta, e os clientes que contratam o empréstimo recebem o dinheiro que necessitam. Se o pagamento da taxa de juros fosse feito apenas a clientes cujo dinheiro foi utilizado para um empréstimo, aqueles que não têm sua parte utilizada com o tempo deixariam de fazer parte do serviço.

Por isso, diferente de uma abordagem empresarial onde os recursos são gerenciados como um todo, quando se abre a possibilidade para que agentes externos se conectem ao serviço e por isso sejam recompensados financeiramente, estes esperam que, muito ou pouco, recebam algo pela disponibilidade dos seus equipamentos, e neste ponto, se sempre os mesmos 70% dos recursos forem utilizados, com o tempo os 30% restantes deixarão de oferecer seus equipamentos na plataforma, causando uma queda do total de recursos computacionais disponível no serviço.

Levando em consideração que um serviço como este pode chamar pessoas a compartilhar tanto recursos de alta capacidade, como outros que já estavam sem uso em virtude de uma substituição por um equipamento mais moderno, e por isso não têm a mais alta capacidade computacional, se estes forem colocados a disposição no ambiente sem adequação de uma regra, os equipamentos de maior capacidade acabariam por atender a maioria, se não todas, as demandas, tornando novamente inúteis os equipamentos mais simples, o que seria uma repetição do mesmo problema que já se instala atualmente.

Com esta perspectiva, a utilização dos recursos deve ser realizada com critério, por isso não se pode apenas destinar as solicitações primeiramente aos recursos de maior capacidade, pois assim sempre aqueles com menor capacidade serão os últimos a serem utilizados, gerando a fuga dos provedores de recursos da arquitetura pela falta de utilização.

Como os recursos provedores podem também ser de utilização constante pelos próprios que utilizam a arquitetura, é necessário um conhecimento mais aprofundado do recurso, não considerando apenas a capacidade de processamento como o único fator determinante para se identificar qual o seu poder computacional, portanto além de saber qual é a composição de cada um deles, se faz necessário o uso de um mecanismo de detecção do momento mais oportuno para a sua utilização, assim como uma identificação mais completa do seu poder computacional.

Demandas nem sempre possuem os mesmos tipos de requisitos para sua execução, em alguns casos a capacidade de processamento pode ser um fator determinante para a escolha, porém, podem haver casos em que a atividade demanda uma quantidade de memória acima do que se é requisitado normalmente, por isso a capacidade de processamento pode se tornar um item secundário na escolha. A capacidade total de um recurso é melhor avaliada quando mais de um componente é levado em consideração, por isso existe aqui a preocupação de equalizar o poder computacional total de um recurso permitindo que se conheça melhor qual é a sua capacidade, envolvendo mais de um componente interessante ao processo.

Neste estudo foi definida uma arquitetura capaz de suprir estas carências citadas anteriormente, possibilitando uma proporcionalidade de uso de todos os recursos computacionais envolvidos no processo, aliado a disponibilização de um mecanismo para estabelecer o comportamento padrão de cada recurso que, por sua vez, ajudará na identificação do recurso mais propício para o recebimento da demanda solicitada.

Para atingir tal especificação, a arquitetura se organiza em algumas etapas, iniciando pela construção de um valor que melhor identifique o poder computacional de um recurso, permitindo que seja avaliado de acordo com as demandas que o ambiente alvo necessitam. Tendo

o poder de cada recurso definido, uma segunda etapa se organiza, onde estes são dispostos em grupos, de forma que não sejam mais analisados isoladamente e, evitando que recursos de baixo poder computacional tenham sua escolha rejeitada em virtude da frequente escolha de recursos com alto poder computacional. O objetivo aqui é permitir um maior engajamento destes, evitando assim que deixem a arquitetura por falta de uso. Cada recurso é monitorado individualmente, e a partir disso é construída uma Assinatura Comportamental, a qual representa o comportamento de um recurso durante o dia normal de funcionamento. Todas as Assinaturas Comportamentais dos recursos de um *cluster* são integradas formando a Assinatura dele, o que viabiliza que um *cluster* seja escolhido de acordo com a sua capacidade de contribuir em um determinado momento com a demanda.

Isso apenas resume o funcionamento para que se entenda a ordem de execução das etapas, porém um aprofundamento da arquitetura é detalhado mais a frente, permitindo uma melhor compreensão de como funcionam e quais métodos cada etapa utiliza.

4.1 METODOLOGIA

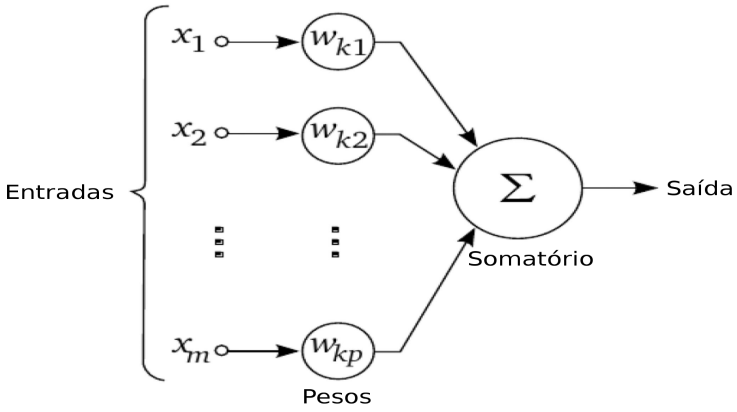
Para que a definição da característica e da capacidade de cada recurso computacional não se restrinja ao seu poder de processamento, é necessário que mais de um componente do recurso seja levado em consideração. Porém, pode-se dizer que a contribuição executada por cada um destes componentes varia, visto que alguns são mais relevantes que outros e por isso um aumento em um determinado componente pode representar uma melhora significante no poder computacional total.

Ainda existe a variação de que cada cenário, ou cada ambiente de aplicação do método, pode ter uma realidade que exija mais de um componente, aumentando assim a relevância deste no âmbito do poder computacional. Existem ferramentas para execução de *benckmarks*, os quais avaliam a performance e escolhem *upgrades* efetivos em um equipamento, porém estas ferramentas fazem estas avaliações seguindo um único padrão, por isso não se ajustam ao sistema em que o equipamento está empregado. Deste modo, apesar de serem ferramentas muito difundidas no meio da ciência da computação, os *benckmarks* não permitem o ajuste referido anteriormente, o qual pode significar uma mudança no cálculo e resultado do poder computacional do recurso.

Para que se possa obter um cálculo do poder computacional que se ajuste de acordo com a necessidade do ambiente alvo, se propõe utilizar um conceito de redes neurais nomeado como "perceptron", o qual é o modelo mais simples de representação de um neurônio, e pode ser visto na figura 9.

Neste modelo as entradas, que são características ou parâmetros informados de acordo com a sua aplicação, são submetidas ao modelo perceptron, cada uma com um peso correspondente. Na figura 9 os elementos x são os parâmetros enviados ao modelo, enquanto wk são os seus pesos respectivos. Tendo recebido todos os dados necessários, o modelo executa um

Figura 9 – Modelo de um neurônio artificial.



Fonte: Adaptado de (HAYKIN, 2008)

processamento que resulta em uma saída.

No caso da aplicação executada neste trabalho, as características representadas pelos elementos x serão os componentes do recurso computacional, como CPU, memória RAM ou velocidade da rede, e os pesos referenciados por wk poderão ser ajustados conforme a necessidade de cada ambiente de aplicação do modelo. O processo referenciado na figura, e designado como somatório, será efetivamente a soma dos valores das características multiplicadas pelo seu peso associado. A definição pode ser melhor compreendida pela equação 4.1 abaixo:

$$\sum_{i=0}^{m} Pi * Ci \tag{4.1}$$

Neste cálculo, P representa o peso respectivo de cada componente (CPU, Memória RAM, Latência, etc...) relevante do recurso para o ambiente aplicado e C a sua quantidade (3GHz, 4Gb, 100ms) presente. É através do valor obtido por este cálculo, definido como poder computacional, que se é caracterizado o quando o recurso pode contribuir com a arquitetura montada.

Em um ambiente de grade computacional é comum que os recursos de maior poder de processamento sejam os que mais recebem solicitações de execução de tarefas, justamente por a capacidade de processamento ser um item predominante na escolha do recurso executor. Da mesma forma, se unicamente for utilizado o poder computacional calculado anteriormente, e priorizado o envio das tarefas a aqueles que possuem um poder maior, mesmo que o processo leve em consideração mais componentes do recurso e não só a capacidade de processamento, o resultado será o mesmo, visto que os elementos de maior poder computacional ainda seriam os mais requisitados, deixando para utilizar aqueles com menor poder somente em casos que os recursos principais estejam sobrecarregados.

Em uma situação ideal todos os recursos teriam a mesma capacidade de participar do processo, tendo assim o mesmo poder computacional e participando com a mesma ênfase das requisições de processamento, porém, casos em que os recursos que integram a arquitetura pos-

suem um poder computacional equivalente e recebem aproximadamente a mesma quantidade de solicitações são raros, o mais comum é que recursos das mais distintas características se agrupem nela. Também não se pode levar em consideração o balanceamento através do rearranjo de componentes internos entre os recursos para equalizar o seu poder computacional, visto que estes podem estar em ambientes ou mesmo continentes diferentes, impedindo uma manobra em tempo hábil para equilibrar o poder total.

Equilibrar o poder computacional dos recursos não é algo que se possa obter facilmente, portanto neste trabalho uma abordagem para este tipo de situação é sugerida em (SENGER; GOIS, 2017a), o que permite uma aproximação deste equilíbrio buscado. Nele é citado que se comparados diretamente uns aos outros, os recursos apontam diferenças entre suas capacidades, porém se grupos forem formados unindo recursos distintos, ao final é possível obter grupos cuja soma dos poderes computacionais é equivalentes entre eles.

Estes grupos, neste trabalho denominados como *clusters*, podem ser formados com a utilização de vários algoritmos consagrados, já testados e aplicados nas mais diversas áreas, alguns deles foram citados e analisados conforme suas diferentes formas de atuação. Porém, algoritmos como o *k-means* agrupam os elementos por similaridade, o que no final do processo geraria *clusters* dos recursos com maior poder de processamento ao lado de outro com os recursos de menor poder, aumentando ainda mais a diferença entre eles.

A ideia é criar grupos com poderes computacionais equivalentes uns aos outros, permitindo que ao serem comparados durante o processo de destino de uma requisição, qualquer um tenha a mesma capacidade de atender a demanda. O processo é o detalhado em (SENGER; GOIS, 2017a) e, pode ser melhor compreendido na figura 10.

Como pode ser visto no processo demonstrado na figura 10, cada recurso computacional possui a sua característica de capacidade de CPU, memória, latência entre outras, e assim as mais relevantes para o processo são submetidas ao sistema de determinação do poder computacional baseado no modelo perceptron.

Para demonstrar a utilização do modelo, considerando que em um determinado ambiente em que foi aplicado, as características dos recursos tenham os pesos demonstrados na tabela 2 a seguir:

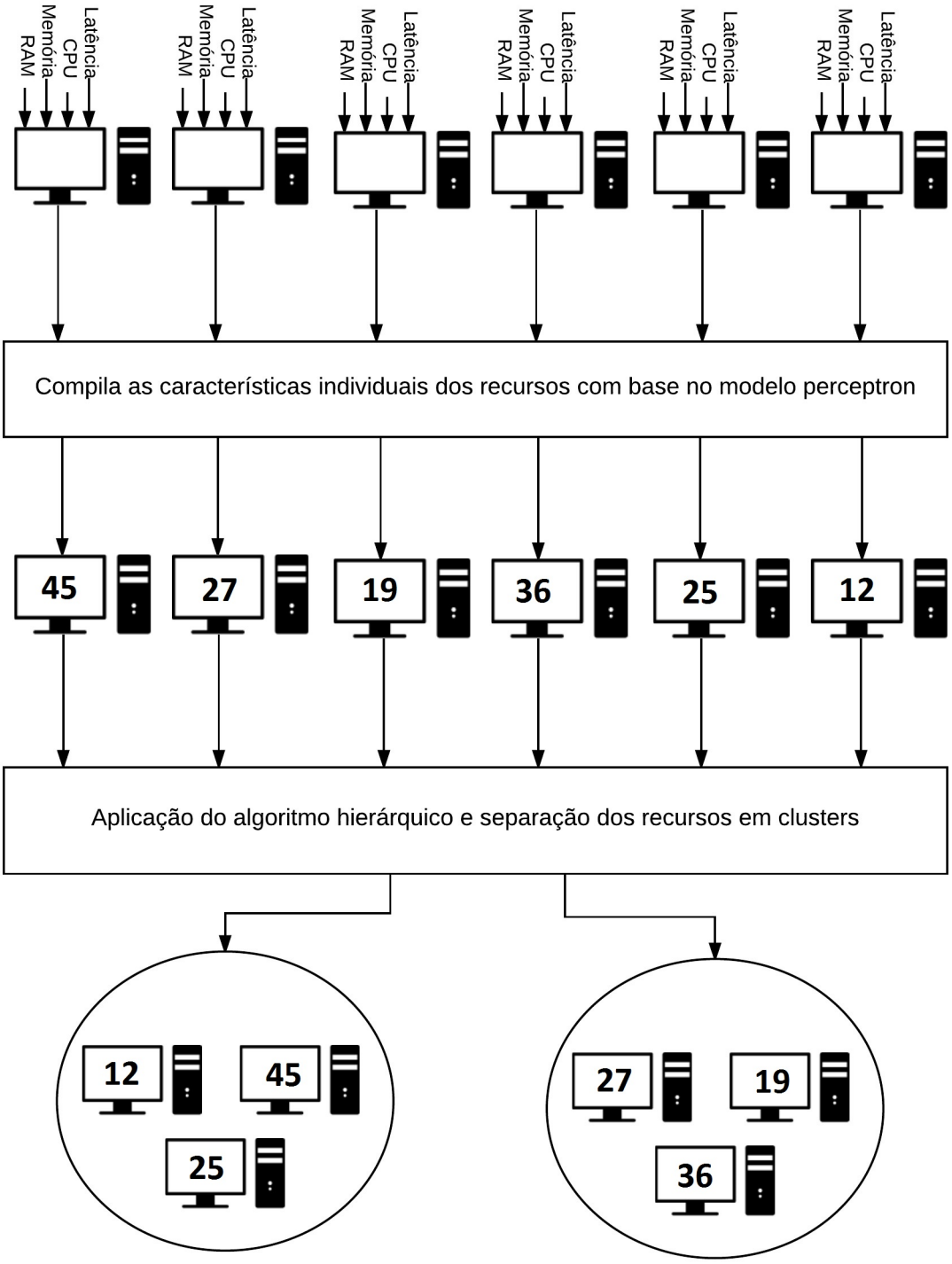
Tabela 2 – Peso dos componentes

Peso dos Componentes				
Frequência	Físicos	Lógicos	RAM	Latência
4	4	3	2	-0.05

Fonte: O Autor

Segundo o que consta na tabela 2, os componentes frequência e núcleos físicos são os que mais impactam na execução, sendo atribuído a eles os maiores valores de peso. Algo a se salientar também é a existência de um peso com valor negativo, isso se deve ao fato de que esta é uma grandeza inversamente proporcional, ou seja, quanto maior seu valor mais prejudicial para o ambiente este componente é, se aproximando do ideal quanto mais próximo de 0 estiver

Figura 10 – Clusterização de recursos computacionais.



Fonte: (SENGER; GOIS, 2017a)

o índice do componente. Os valores de maior peso neste exemplo denotam que as solicitações enviadas a ele costumam exigir muito processamento, por isso um sistema com maior número de núcleos e maior frequência é mais apropriado para a aplicação. Estes valores podem ser alterados e ajustados pelo responsável pela aplicação do sistema, de acordo com a sua percepção

da necessidade no ambiente.

Por se tratar apenas de uma expressão de grandeza, é possível que em determinado cálculo um recurso possua poder computacional negativo em virtude dos pesos também negativos aplicados no ambiente. Este valor sendo negativo não traz nenhum problema, é apenas uma expressão da capacidade do recurso perante o ambiente em que está inserido.

Tabela 3 – Exemplo de configuração de equipamentos

Equip.	Freq.	Núc. Físicos	Núc. Lógicos	RAM	Latência
1	2.3GHz	2	2	8	64ms
2	3.2GHz	1	1	4	56ms
3	3.2GHz	3	3	6	16ms
4	2.2GHz	4	4	8	516ms
5	2.5GHz	1	1	2	40ms
6	1.5GHz	2	2	2	240ms

Fonte: O Autor

Na tabela 3 é demonstrado alguns recursos e seus respectivos valores para cada componente, que após análise teriam o seu poder computacional definido. O resultado do valor correspondente a cada elemento pode ser visualizado e melhor compreendido na tabela 4 onde em cada elemento com o valor convertido para o peso correspondente e após isso somado a todos os outros, é computado e integrado ao poder computacional referido na última coluna. Após processados pelo modelo proposto estariam dispostos segundo consta na figura 10.

Tabela 4 – Exemplo de configuração de equipamentos

Equip.	Freq.	Núc. Físicos	Núc. Lógicos	RAM	Latência	Poder Comp.
1	9,2	8	6	16	-3,2	36
2	12,8	4	3	8	-2,8	25
3	12,8	12	9	12	-0,8	45
4	8,8	16	12	16	-25,8	27
5	10	4	3	4	-2	19
6	6	8	6	4	-12	12

Fonte: O Autor

Pensando em uma situação futura onde os peso de cada componente possa ser ajustado, ou por um administrador ou por uma análise do próprio sistema relativo as demandas recebidas, o cálculo do poder computacional de cada recurso não é feito internamente a ele, mas dentro do recurso que gerencia toda a estrutura. Desta forma não é necessária mais uma comunicação com cada recurso para que seja refeito o cálculo do poder computacional após cada ajuste na arquitetura.

4.1.1 Clusterização

Após determinado o poder computacional, estes recursos passam por um processo de clusterização, garantindo uma homogeneidade nos agrupamentos destes recursos em *clusters* com capacidades computacionais semelhantes.

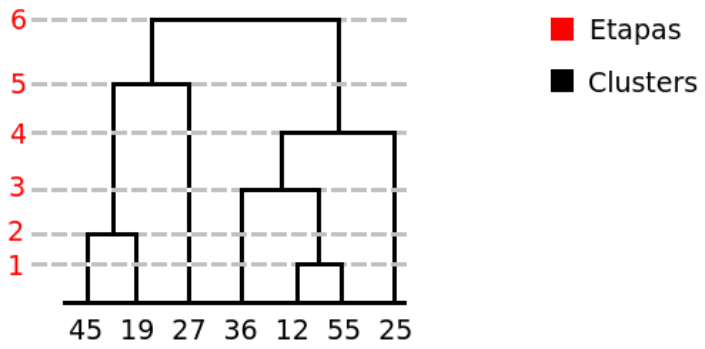
Para que os *clusters* gerados sejam equivalentes, é utilizado o algoritmo hierárquico, o qual além de se diferenciar de algoritmos clássicos como *k-means* que agrupam por similaridade, possui a vantagem de não ser necessário que o sistema determine qual o número de *clusters* que se deseja gerar, assim permitindo ao sistema determinar qual o número que melhor representa a separação que lhe é interessante.

A aplicação do Algoritmo Hierárquico de início levantou uma questão que acaba por ser recorrente em processos de clusterização, que é: Qual o número ideal de *clusters*? Como este algoritmo possui um formato diferente de clusterização de outros como *k-means*, onde é necessário de início se identificar qual o número exato esperado de *clusters*, foi necessário adaptar o resultado a realidade da aplicação.

O resultado final de um Algoritmo Hierárquico é um gráfico de Dendograma, e cada junção nova nele significa a união de dois novos *clusters*. Ao final do processo o que se obtêm é algo próximo a um histórico de todas as junções e assim portanto, de todos os *clusters* formados pelo algoritmo, iniciando em uma etapa onde todos possuíam apenas um elemento, até o final onde resta apenas um *cluster* contendo todos os elementos do processo.

Possuindo o resultado final da clusterização, é possível avaliar após cada etapa como ficam os *clusters*, definindo assim em qual etapa existe um balanceamento mais coeso. Observando a figura 11 é possível se perceber melhor os elementos agrupados a cada etapa.

Figura 11 – Dendograma e níveis da clusterização.



Fonte: O Autor

As etapas na figura 11 demonstram quais os elementos que passam a compor os grupos após cada agrupamento. Antes da etapa 1, todos os elementos estão contidos em grupos com um único elemento, no caso eles mesmos. O valor dos grupos é a soma do valor de seus elementos, por isso na primeira etapa o valor do grupo é o valor do seu único elemento. Após a primeira etapa os *clusters* de valores 12 e 55 são unidos, passando então a existir um *cluster* de valor total 67, permanecendo então em ordem os valores 19, 25, 27, 36, 49 e 67.

Como a busca neste estudo é por grupos balanceados, no resultado do Algoritmo Hierárquico desconsideramos as etapas onde existam ainda *clusters* com apenas um elemento. Neste exemplo a etapa onde isso deixa de ocorrer é na 5. Porém como a sexta etapa encerra o processo unindo todos em um mesmo *cluster*, a etapa 5 passa a ser a única possível de ser utilizada, fina-

Tabela 5 – *Clusters* do Dendograma de 50 elementos da figura 12

Etapa.	Menor *Cluster*	Maior *Cluster*	Distância
38	47.0	130.0	83.0
39	47.0	158.0	111.0
40	68.0	158.0	90.0
41	96.0	160.0	64.0
42	96.0	240.0	144.0
43	111.0	240.0	129.0
44	111.0	324.0	213.0
45	111.0	324.0	213.0
46	160.0	351.0	191.0
47	160.0	649.0	489.0
48	160.0	973.0	813.0
49	1133.0	1133.0	0.0

Fonte: O Autor

lizando o processo com os *clusters* e elementos: 1 (45, 19, 27); 2 (36, 12, 55, 25); com valores totais de 91 e 128.

Em situações com quantidades menores de elementos esta ocorrência é comum, porém quando o número de elementos sobe, também aumenta a quantidade de agrupamentos, e o momento em todos os *cluster* deixam de ser composto por apenas um elemento ocorre várias etapas antes do término, necessitando assim de uma análise sobre os resultados para se definir qual a melhor relação de *clusters* disponibilizada pelo algoritmo.

Na figura 12 ocorre um processo de clusterização com 50 elementos. Este dendograma foi alterado para que na esquerda em vermelho sejam exibidos não a altura da junção como acontece em um gráfico comum deste tipo, mas sim para demonstrar a etapa em que cada uma das junções dos *clusters* acontece.

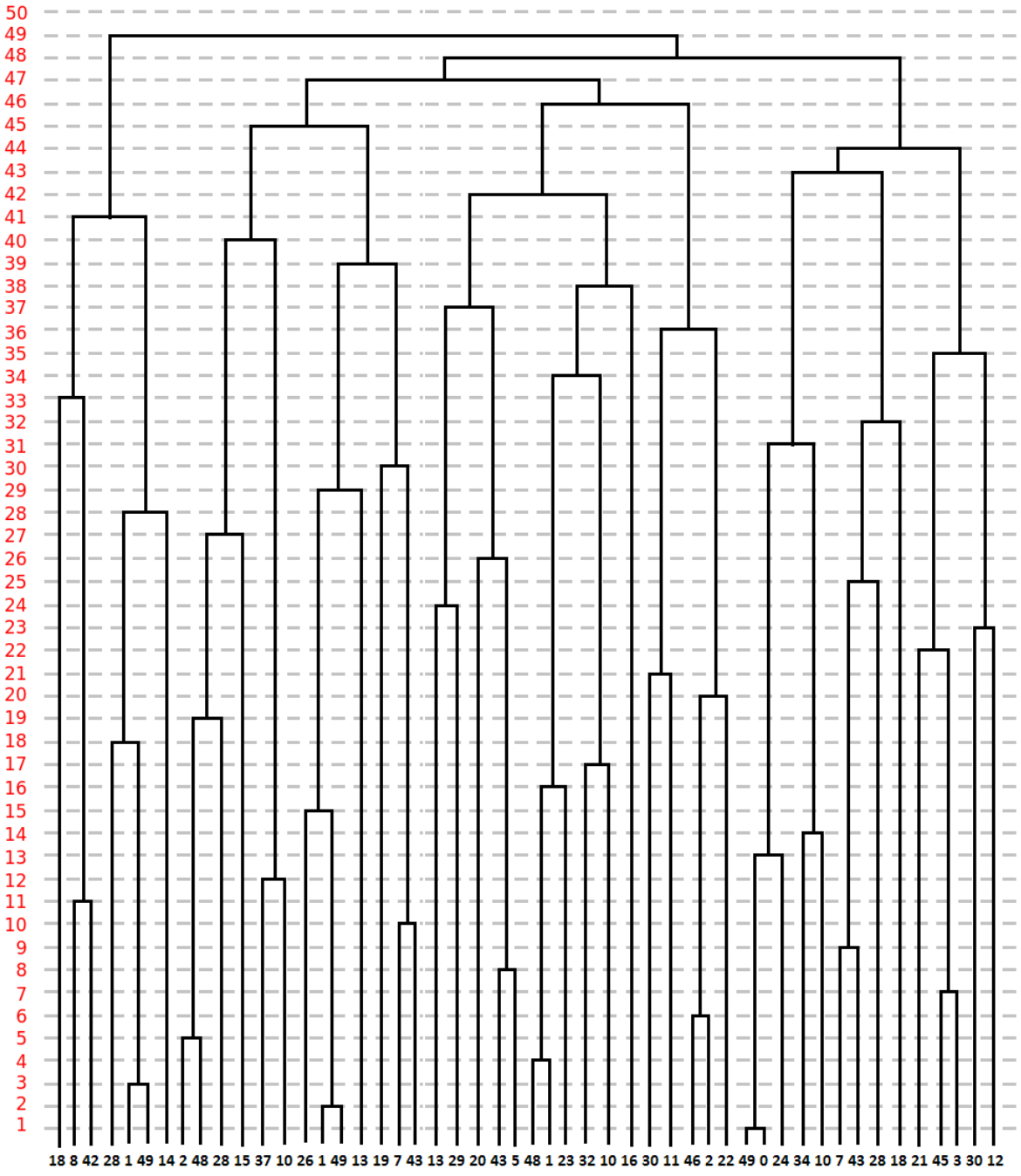
No dado exemplo, o último *cluster* a possuir apenas um elemento é unido a outro na etapa 38, portanto deste ponto até a penúltima etapa de clusterização é que os *clusters* serão analisados.

Possuindo os *clusters* a serem analisados, verificamos a distância entre a soma dos elementos do maior e do menor *cluster*, o que pode ser melhor visualizado na tabela 5.

O *clusters* da etapa 49 têm distância 0 pois esta é a etapa final, onde apenas um *cluster* permanece, portanto somente as etapa de 38 à 48 serão analisadas. Observando a distância entre o maior e menor grupo, aquele que apresenta o menor valor para esta coluna é o agrupamento gerado na etapa 41, onde a distância é de 64. Portanto nesta situação, dentre todos os agrupamentos possíveis gerados pelo algoritmo, o resultado mais balanceado gerado são com os elementos demonstrados na tabela 6

É importante ressaltar que, como o algoritmo utiliza um método matemático para cálculo da junção entre os *clusters*, seu resultado é sempre o mesmo independente da quantidade de vezes que for executado, portanto, apesar de em certos casos ser perceptível que um refinamento pode ser aplicado, a resposta encontrada é sempre a melhor possível para o método do algoritmo.

Figura 12 – Dendograma com 50 elementos.



Fonte: O Autor

Tabela 6 – *Clusters* com melhor balanceamento para o Dendograma de 50 elementos

Clusters	Elementos	Número de Elementos	Valor Total
1	2, 48, 28, 15, 37, 10,	6	140
2	13, 29, 20, 43, 5,	5	110
3	48, 1, 23, 32, 10, 16,	6	130
4	26, 1, 49, 13, 19, 7, 43,	7	158
5	49, 0, 24, 34, 10,	5	117
6	21, 45, 3, 30, 12,	5	111
7	30, 11, 46, 2, 22,	5	111
8	7, 43, 28, 18,	4	96
9	18, 8, 42, 28, 1, 49, 14,	7	160

Fonte: O Autor

4.1.2 Assinatura Comportamental do Recurso

O monitoramento instantâneo do uso dos componentes do recurso, pode não representar a sua situação exata no momento da execução da demanda. Como o uso total oscila frequentemente de acordo com o tempo, é possível que um equipamento momentaneamente apresente um baixo uso, e logo após tenha este uso elevado em virtude de atividades cotidianas normais atribuídas a ele, tornando a execução e resposta lenta.

Gráfico 5 – Gráfico de uso atual de recursos computacionais.

Fonte: O Autor

A situação citada anteriormente pode ser melhor compreendida observando o gráfico 5, a qual demonstra no $Recurso\ A$ na área demarcada um uso de quase 90%, enquanto no $Recurso\ B$ este se encontra em aproximadamente 25%. Uma análise instantânea da situação atual do recurso denotaria que o $Recurso\ B$ estaria em melhor condição de receber uma solicitação do que o $Recurso\ A$, pois no momento avaliado ele apresenta uma ociosidade maior.

Porém se o comportamento destes recursos fossem analisados periodicamente, seria possível prever como seria o seu uso para um momento a frente desta etapa de leitura momentânea, o que pode ser melhor visualizado na figura abaixo.

No gráfico 6 é possível perceber que o $RecursoA$ apesar de estar no momento com

Gráfico 6 – Gráfico de uso conhecido de recursos computacionais.

Fonte: O Autor

um uso maior que o $RecursoB$, logo após ele diminui para aproximadamente 25%, enquanto que o $RecursoB$ sobe para 80%, demonstrando um pico momentâneo no primeiro, assim como uma queda no segundo. Sendo assim, se o escalonador tiver o conhecimento prévio da Assinatura Comportamental destes recursos, uma decisão mais acertada seria tomada no momento do direcionamento da demanda.

A elaboração desse gráfico pode ser conseguida através de mecanismos de predição de comportamento do recurso, um deles é a Assinatura Comportamental. Através do monitoramento do recurso este padrão pode ser obtido e com o tempo ele é refinado até se aproximar ao máximo da realidade. Este método é utilizado também para a identificação de anomalias, que podem ser falhas ou uso fora do comum.

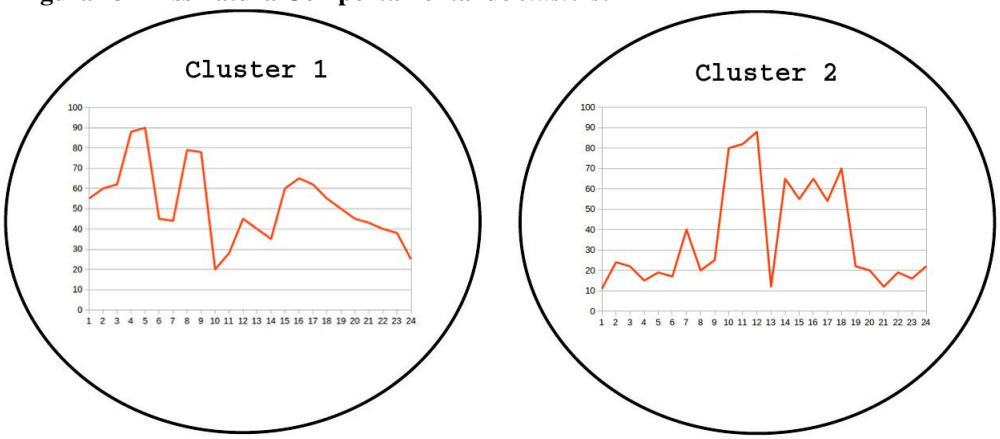
Dentro do método criado neste projeto, cada recurso passa a ser monitorado e sua Assinatura construída, de modo que possa a se conhecer o funcionamento de cada um dos componentes de cada *cluster* do ambiente. Não somente o recurso, mas também o *cluster* passa a ser identificado através de sua Assinatura Comportamental, elaborada a partir da média de todas as assinaturas dos recursos agregados a ele.

Os *clusters*, visualizados de fora, passam a ter a Assinatura Comportamental como uma característica própria, e podem ser analisados tal qual demonstrado na figura 13. Esta assinatura global do *cluster* composta pelas assinaturas dos seus recursos pode ser utilizada como parâmetro para definição do destino de uma requisição. Já que o poder computacional dos *clusters* é equivalente, a assinatura do uso do *cluster* passa a ser um diferencial para a escolha e determinação do recurso mais adequado no momento.

Internamente os *clusters* são agrupamentos de recursos, cada um com a sua Assinatura Comportamental, que juntos compõe a Assinatura do *cluster* ao qual eles fazem parte. Esta composição pode ser melhor compreendida visualizando a figura 14.

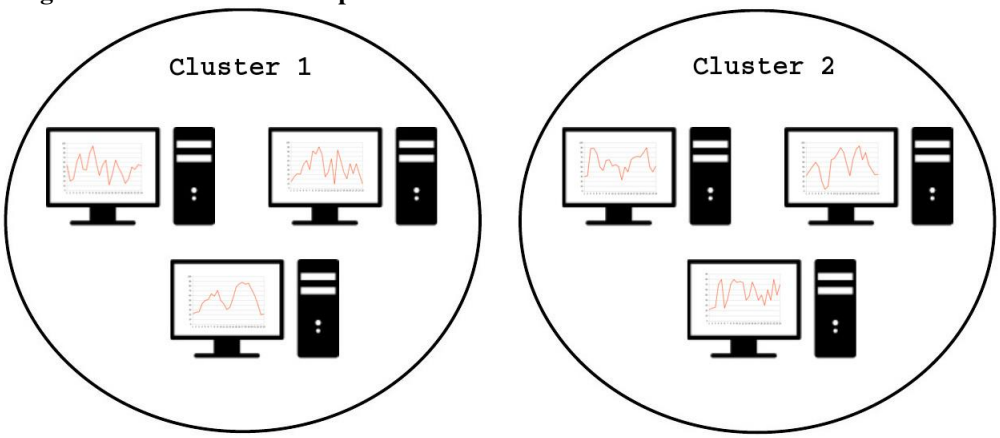
A composição desta estrutura permite por fim homogeneizar a escolha dos recursos para execução das tarefas, aumentando as chances de escolha de um recurso com menor capacidade, visto que externamente esta escolha fica sujeita mais a assinatura do *cluster* do que propria-

Figura 13 – Assinatura Comportamental de *clusters*.



Fonte: O Autor

Figura 14 – Assinatura Comportamental dos recursos dos *clusters*.



Fonte: O Autor

mente à capacidade de processamento do recurso computacional, cumprindo com o objetivo da arquitetura.

Para composição das assinaturas dos recursos computacionais será aplicado o método descrito em (SENGER; GOIS, 2017b), o qual faz uso do algoritmo de clusterização *k-means* pra análise do padrão de cada intervalo de tempo que compõe a assinatura do recurso.

A assinatura construída pelo método possui uma margem natural de execução do recurso, como pode ser melhor compreendida na figura 3, onde a linha em vermelha se refere ao limite superior de uso natural, e a linha em azul ao limite inferior, sendo que a área em verde é compreendida como onde o recurso costuma estar durante um período de funcionamento normal.

4.1.3 Assinatura Comportamental do *Cluster*

Para a composição da assinatura do *cluster*, o método aplicado será diferente daquele de um único recurso. O motivo para isso é que para cada intervalo de tempo analisado do recurso,

podem haver dezenas e até centenas de variáveis de dados a serem dispostas para composição de sua assinatura neste dado período, inclusive, vários dos pontos analisados podem não fazer parte da assinatura final devido a natureza dos métodos de criação, os quais utilizam todos os pontos lidos para composição da assinatura, com isso, após análise, nem todos acabam fazendo parte do resultado, visto que algumas leituras podem não representar a execução natural do recurso, e assim, deixando de fora os *outliers*, que podem ser analisados mais tarde por outro método como uma anomalia.

Diferente da assinatura dos recursos onde todos os dados são lidos, mas nem todos são a representam no final, as assinaturas dos *clusters* fazem uma representação de todos os recursos que estão imersos nele, e por isso nenhum dos recursos pode ficar de fora da assinatura final. Neste sentido, não é possível aplicar o mesmo método por dois motivos:

- O volume de dados analisado é muito menor que o utilizado para composição da assinatura do recurso;
- Todos os recursos devem ter sua parcela de contribuição para composição da assinatura do *cluster*.

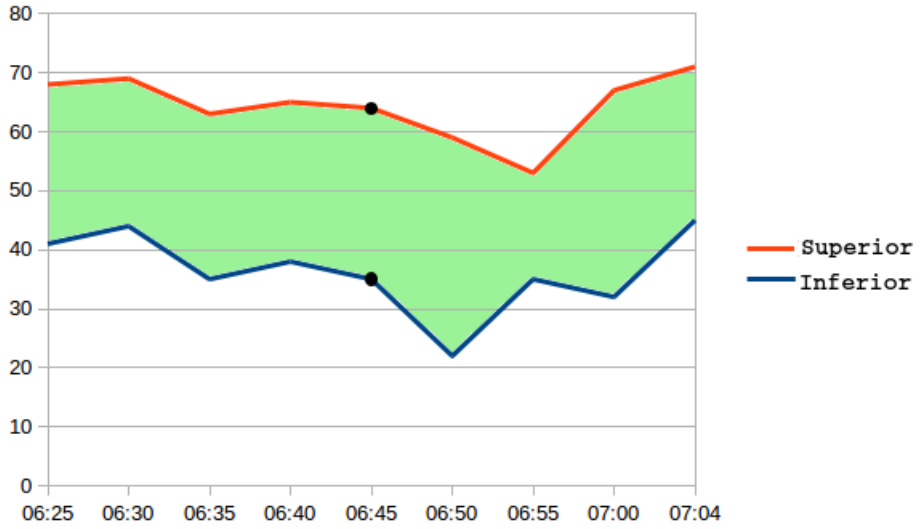
Se todos os recursos devem ter sua parcela de contribuição, nenhum recurso pode ser descartado da assinatura comportamental do *cluster*, não permitindo que um deles não tenha representação no resultado final, como acontece na assinatura do recurso, o qual despreza algumas leituras consideradas como fora do padrão. Portanto, mesmo que a contribuição seja em uma proporção inferior à de outros recursos, pelo motivo de todos fazerem parte do *cluster*, todos ficam representados.

Assim como o poder computacional do *cluster* é composto pela soma dos recursos, e logicamente um recurso com maior poder impacta mais no crescimento do poder do *cluster*, da mesma forma deve ser impactada a assinatura do *cluster* pelo recurso que mais contribui com ele. Deste modo, para construção da assinatura do *cluster*, é analisado o mesmo intervalo de tempo de cada um dos recursos, considerando o ponto superior da sua assinatura comportamental, e também o ponto inferior, assim como demonstrado no gráfico 7, onde estão demarcados pontos na linha de limite superior e inferior:

O gráfico 7 demonstra uma assinatura fictícia de um suposto recurso computacional, e nela dois pontos são destacados, os quais se referem os limites superiores e inferires, delimitando o intervalo comum de funcionamento do recurso em um horário específico, que no caso é o das 06:45h. Para cada recurso imerso no *cluster*, é analisado esses dois pontos no mesmo intervalo de horário.

Como citado anteriormente, os recursos contribuem proporcionalmente ao crescimento do poder computacional do *cluster*, por isso estes limites superiores e inferiores também tem um peso proporcional na construção da sua assinatura. Para isto os limites são submetidos a um cálculo utilizando uma média ponderada, onde os valores são os números presentes na assinatura

Gráfico 7 – Assinatura Comportamental com pontos em limite superior e inferior demarcados.



Fonte: Adaptado de (SENGER; GOIS, 2017b)

de cada recurso, e o peso é o poder computacional de cada. Desta forma os recursos que mais impactam na capacidade computacional do *cluster*, também impactam na mesma proporção na assinatura.

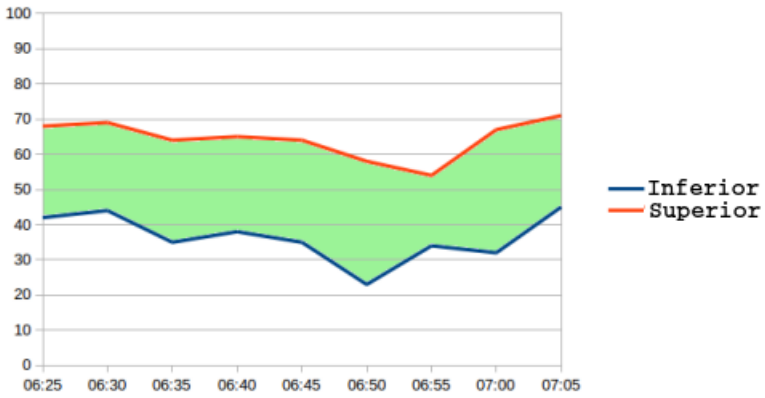
Para cada um dos limites é executado o mesmo cálculo separadamente, portanto para o limite superior são coletados todos os valores dos recursos computacionais naquele intervalo de tempo, da mesma forma para os limites inferiores.

Após o cálculo, dois valores são gerados, um para o limite superior e outro para o inferior, valores esses que delimitarão a margem de funcionamento dos recursos do *cluster*, e assim definem a sua Assinatura Comportamental.

Quando todos os intervalos de tempo forem calculados, então será criada a Assinatura Comportamental do referido *cluster*. Para exemplificar a aplicação do cálculo, nas figuras 15, 16 e 17 são demonstrados alguns recursos computacionais e suas respectivas assinaturas e valores que às compõe, de um dado intervalo de tempo do dia.

Figura 15 – Assinatura Comportamental e seus valores em um intervalo de tempo para o Recurso 1.

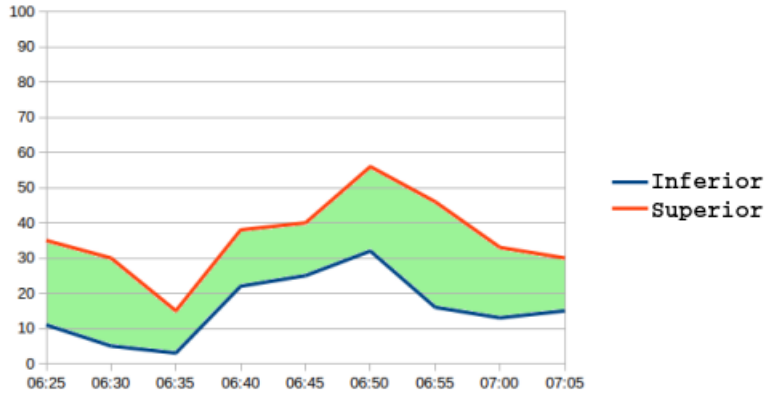
Poder Computacional		87
Hora	Inferior	Superior
06:25	42	68
06:30	44	69
06:35	35	64
06:40	38	65
06:45	35	64
06:50	23	58
06:55	34	54
07:00	32	67
07:05	45	71



Fonte: O Autor

Figura 16 – Assinatura Comportamental e seus valores em um intervalo de tempo para o Recurso 2.

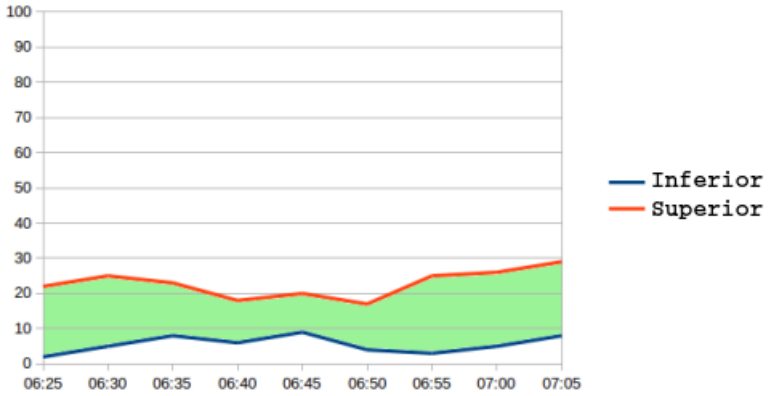
Poder Computacional		189
Hora	Inferior	Superior
06:25	11	35
06:30	5	30
06:35	3	15
06:40	22	38
06:45	25	40
06:50	32	56
06:55	16	46
07:00	13	33
07:05	15	30



Fonte: O Autor

Figura 17 – Assinatura Comportamental e seus valores em um intervalo de tempo para o Recurso 3.

Poder Computacional		55
Hora	Inferior	Superior
06:25	2	22
06:30	5	25
06:35	8	23
06:40	6	18
06:45	9	20
06:50	4	17
06:55	3	25
07:00	5	26
07:05	8	29



Fonte: O Autor

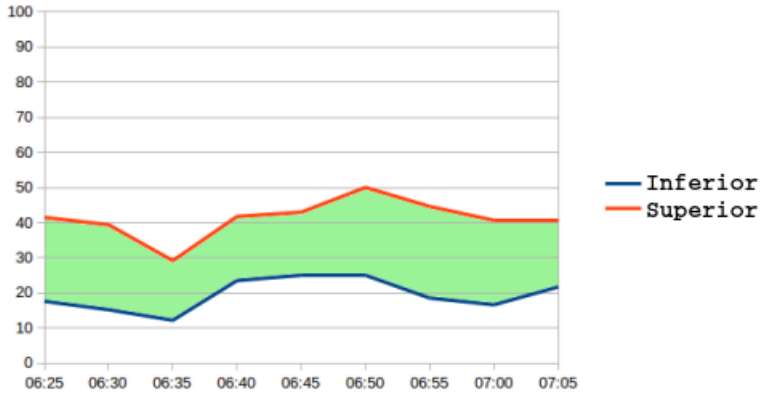
Considerando que os três recursos demonstrados anteriormente estejam dispostos no mesmo *cluster*, para confecção da sua assinatura é necessário utilizar os valores do mesmo intervalo de horário. De início pode-se extrair dos recursos na hora 06:25 os valores 42 e 66 para o Recurso 1, 11 e 25 para o Recurso 2 e 2 e 22 para o Recurso 3. Os poderes computacionais de cada um, 87, 189 e 55 tornam-se os respectivos pesos para o cálculo.

Calculadas separadamente a média ponderada dos valores dos limites inferiores e superiores do referido horário, os valores gerados são 18 e 42 respectivamente. Quando os valores para todos os intervalos de tempo forem calculados, a Assinatura Comportamental gerada e seus respectivos valores é o que pode ser visto na figura 18.

Concluída a construção da Assinatura Comportamental do *Cluster* o processo de confecção da arquitetura proposta se encerra, deixando assim um ambiente com recursos organizados e gerenciados, disponibilizando para uso a própria assinatura dos elementos a qual pode servir de base para a decisão de destino das tarefas a serem executadas dentro de um ambiente de grade computacional.

Figura 18 – Assinatura Comportamental do *Cluster* a partir dos seus recursos.

Cluster		
Hora	Inferior	Superior
06:25	18	42
06:30	15	39
06:35	12	29
06:40	24	42
06:45	25	43
06:50	25	50
06:55	19	45
07:00	17	41
07:05	22	41



Fonte: O Autor

4.2 CRONOGRAMA

O presente projeto teve seu início no mês de setembro de 2016, e foi dividido em 3 etapas principais, descritas a seguir.

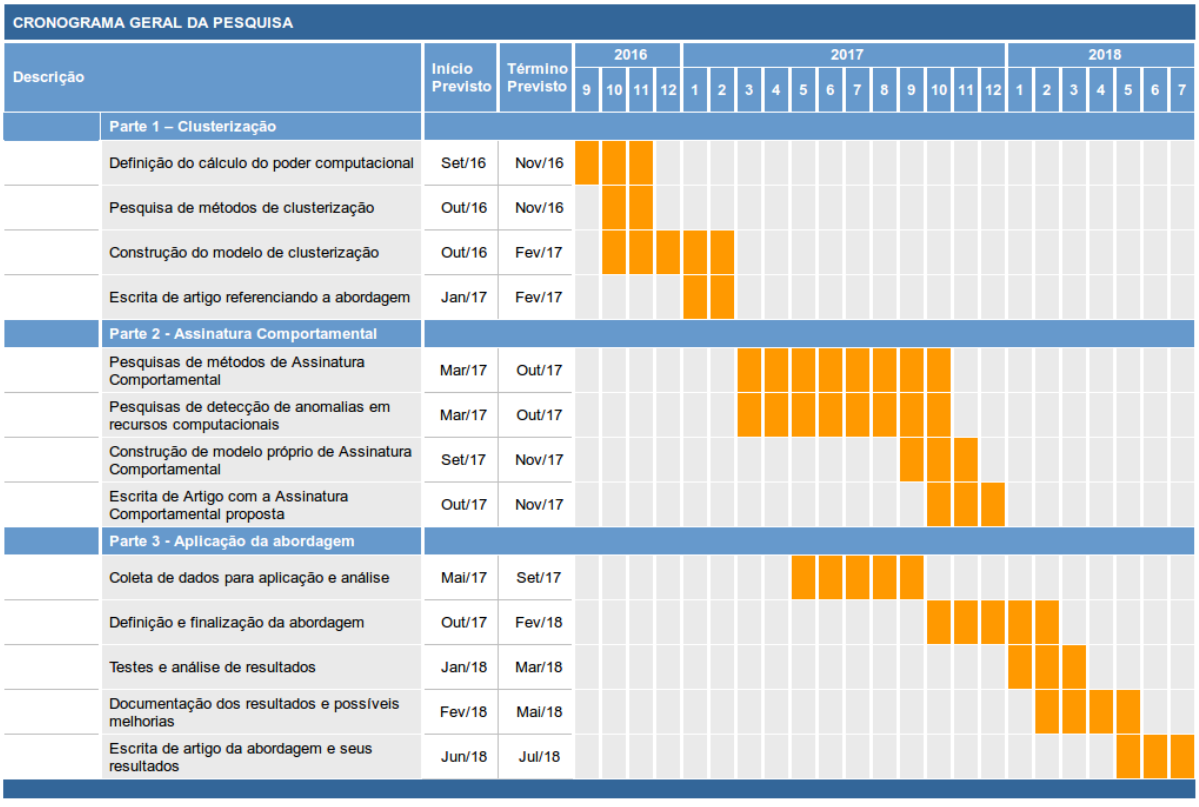
1. **Clusterização** - Foram pesquisadas as formas de clusterização efetivas e também criado um método de cálculo do poder computacional considerando o recurso como um todo, e não somente pela sua capacidade de processamento. Em seguida foi definido o método de clusterização que seria capaz de dispor os recursos em grupos de igual capacidade, tornando-os homogêneos entre si. A partir disto foi possível modelar como seria efetuada a abordagem proposta no projeto, o qual segundo estipulado em cronograma foi escrito, documentado e publicado no artigo (SENGER; GOIS, 2017a).

2. **Assinatura Comportamental** - Como os grupos são homogêneos, uma Assinatura Comportamental adequada faz muita diferença no processo para seleção dos grupos, portanto a sua pesquisa é feita em sequência da definição da clusterização, pois é a partir dos *clusters* que esta assinatura é gerada. Apesar do estudo não envolver detecção de anomalias em si, este método é importante para compreender como as assinaturas são utilizadas no ambiente, onde elas possuem uma melhor aplicabilidade e também onde são falhas. Definida a assinatura o modelo foi descrito em um artigo para publicação.

3. **Aplicação da Abordagem** - Tendo em mãos o modo de clusterização e a forma de geração da Assinatura Comportamental, foi possível construir e testar a metodologia, detectando assim os pontos de melhoria e falhas da abordagem.

A seguir, na Figura 19 é apresentado o cronograma proposto para o controle do andamento do projeto:

Figura 19 – Cronograma de execução da pesquisa.



Fonte: O Autor

5 TESTES E DISCUSSÃO

Pela necessidade de teste com um volume de recursos relativamente grande, por se tratar de uma abordagem a ser aplicada em uma grade computacional, algumas das execuções se utilizaram de simuladores, e quando era permitido, com dados de máquinas reais.

A estrutura foi elaborada utilizando uma aplicação em linguagem java, a qual se utilizou de *sockets* para controlar a conexão entre os recursos e o computador central, denominado de Gerente.

Esta arquitetura tem seu funcionamento baseado em três etapas principais:

1. Clusterização;
2. Assinatura Comportamental do Recurso Computacional;
3. Assinatura Comportamental do *Cluster*;

Nos testes efetuados, uma aplicação executava no Gerente, responsável por controlar e organizar os recursos que viessem a se conectar com a arquitetura. Em cada recurso, uma aplicação fazia o acompanhamento do uso armazenando estas informações a cada 6 segundos em um arquivo texto, separando as informações em arquivos diferentes para cada dia.

Quando a conexão era estabelecida com o Gerente, e o recurso ainda não possuía uma Assinatura Comportamental, ou ainda dados sobre o seu uso ainda não haviam sido coletados. Sendo assim, um identificador aleatório único era gerado para ele, e então colocado em um período de aprendizagem, onde a aplicação nele instalada fazia o seu monitoramento preparando-o para trabalhar no ambiente da arquitetura.

Em se tratando de um recurso já monitorado e com uma assinatura comportamental definida, esta é enviada ao gerente juntamente com seu código único de identificação.

Se o caso fosse de um recurso que já havia frequentado a arquitetura, e sua Assinatura já havia sido construída, ela apenas era reenviada ao Gerente em conjunto com o seu código único de identificação.

Não são somente os recursos recém incorporados a arquitetura que permanecem em aprendizagem sobre o seu uso, todos os recursos são continuamente monitorados. Isto permite o refinamento de todo processo, o que possibilita uma contínua melhoria das assinaturas comportamentais. Outro motivo para isso é a possibilidade deste hábito sofrer mudanças, o que exige uma constante reavaliação e readequação da Assinatura.

Tendo superado a aprendizagem inicial, inicia-se a primeira etapa do processo, onde o recurso é então disponibilizado para alocação em *cluster* segundo oportunidade do Gerente, processo o qual é executado pela clusterização e detalhado a seguir.

5.0.1 Clusterização

Para se testar o processo de clusterização, havia a necessidade da utilização de um número expressivo de máquinas, e para que estas pudessem se adequar na forma de um ambiente totalmente heterogêneo como é habitualmente o de uma grande rede, foi desenvolvido um simulador de conexões, funcionando internamente à aplicação que roda na máquina cliente.

Através de parâmetros de uso foi possível determinar se os dados dos componentes a serem utilizados seriam os que estão instalados na máquina, utilizando assim um modo chamado de "local", ou se deveriam ser gerados novos dados com valores aleatórios, utilizando um modo chamado de "simulador". Quando funcionava no modo simulador, conexões em um intervalo de tempo aleatório eram geradas ao equipamento Gerente, distribuindo também valores aleatórios sobre a quantidade de memória RAM, CPU, número de núcleos da CPU e também uma Assinatura Comportamental fictícia.

Assim foi possível simular a quantidade desejada de máquinas e determinar como a clusterização poderia responder frente aos dados recebidos, pois, para o Gerente não havia nenhuma informação de que as conexões eram provenientes de máquinas simuladas, permitindo que agisse naturalmente à execução da alocação dos recursos.

O primeiro teste foi feito avaliando a resposta do algoritmo, utilizando algumas milhares de máquinas simuladas com valores fictícios de poder computacional. O tempo de execução pode ser visualizado na tabela 7.

Tabela 7 – Tempo de execução do Algoritmo Hierárquico por quantidade de elementos

Quantidade	Tempo (h/m/s)
1000	00:00:04
2000	00:00:34
3000	00:02:06
4000	00:03:59
5000	00:06:41
6000	00:10:30
7000	00:21:14

Para uma quantidade acima de 8000 máquinas o tempo foi superior a 1h de processamento. É possível perceber que mesmo quando se dobra a quantidade de máquinas a se processar, o tempo não é igualmente o dobro, como pode ser visto ao se aumentar a quantidade de 2000 para 4000, onde há um salto de 34 segundos de processamento para 3:59 minutos. Esta diferença se dá pelo formato do algoritmo, que trabalha com uma matriz de distâncias NxN, calculada pela comparação de um a um, dos poderes computacionais de todos os recursos, ainda que os valores utilizados pelo processo sejam equivalentes a (NxN)/2, pois os valores se repetem acima e abaixo da diagonal principal da matriz, a cada iteração esta tabela necessita ser atualizada para que suas distâncias sejam relacionadas ao novo agrupamento gerado. Isto pode ser melhor com-

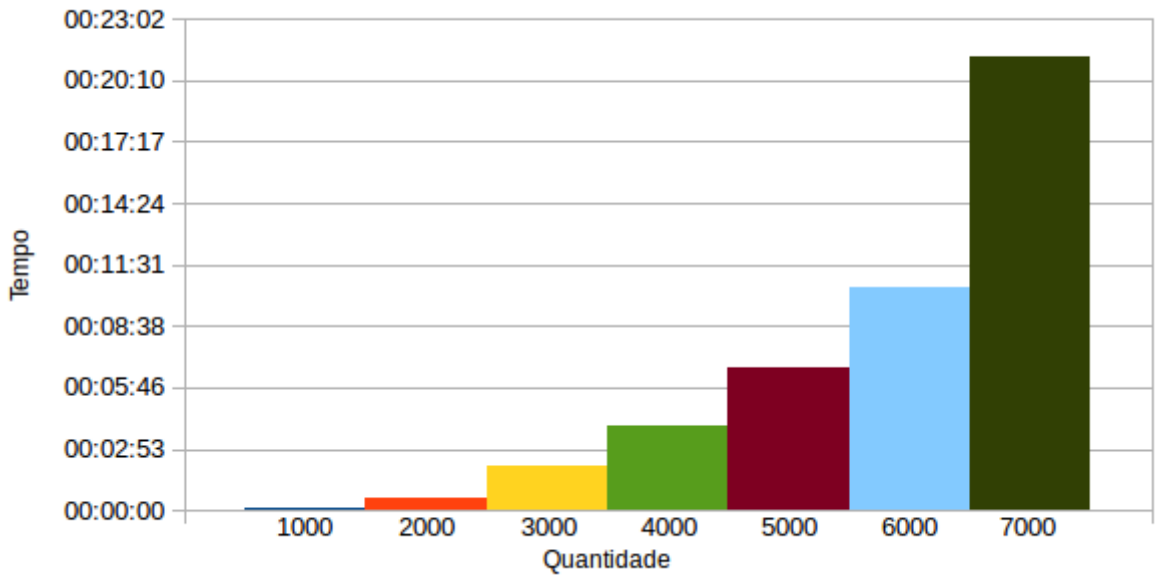
preendido visualizando a tabela 8, a qual representa os poderes computacionais de recursos e a sua distância calculada e, os valores se repetindo acima ou abaixo da diagonal principal.

Tabela 8 – Matriz de Distâncias

PC	15	22	6	9
15	0	7	9	6
2	7	0	18	13
6	9	16	0	3
9	6	13	3	0

Pelas características do algoritmo, impacto do aumento no número de elementos em o dobro não é diretamente proporcional ao tempo de processamento, visto que as operações internas do algoritmo também se tornam mais longas ao se analisar uma quantidade muito maior de dados ja que a montagem das distâncias compõe uma matriz quadrática, por isso o aumento pode não ser exatamente o dobro do tempo.

Gráfico 8 – Tempo de execução do Algoritmo Hierárquico por quantidade de elementos.



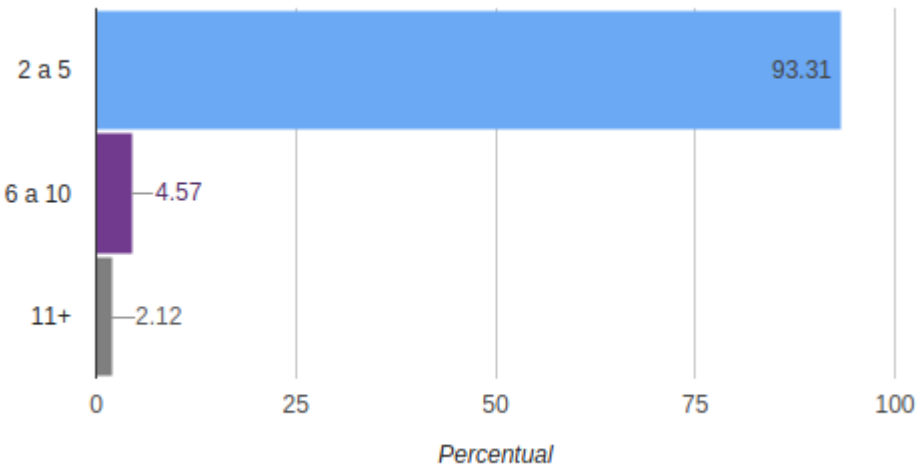
Fonte: O Autor

Os mesmos valores do gráfico 7 podem ser melhor visualizados no gráfico da figura 8, o qual torna mais perceptível o aumento principalmente quando os elementos mudam de 5000 para 6000 e logo após para 7000.

Em relação ao resultado da clusterização, o que pode ser percebido é uma maior facilidade do algoritmo para formar grupos com menor número de elementos, como pode ser visto no gráfico da figura 10.

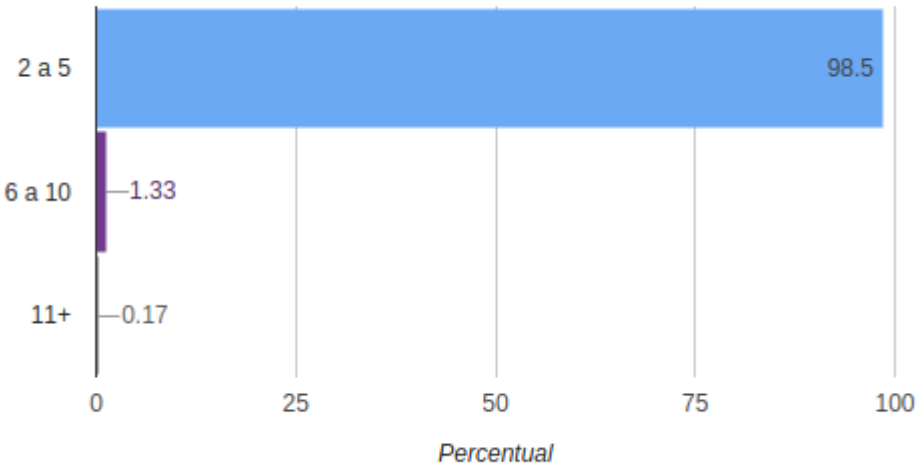
Para situações com poucos elementos a serem clusterizados, a grande maioria dos *clusters* gerados possuía de 2 a 5 elementos, totalizando 93.31% dos casos, sendo que com 50 elementos envolvidos ainda é perceptível a presença de *clusters* com tamanho de 6 a 10 elementos, e apesar de pouco *clusters* com 11 ou mais elementos, também estão presentes nesta situação, como pode ser percebido no gráfico 9.

Gráfico 9 – Percentual de clusters gerados com 50 elementos.



Fonte: O Autor

Gráfico 10 – Percentual de clusters gerados com 100 elementos.



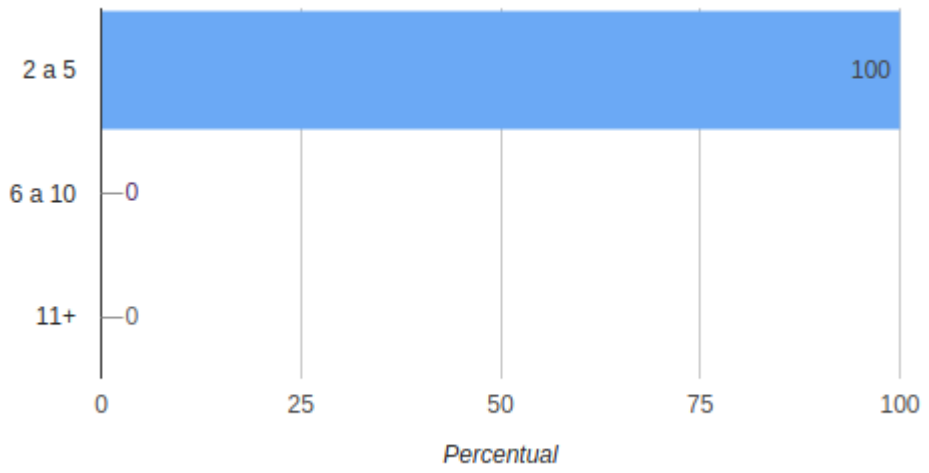
Fonte: O Autor

A medida que o número de elementos a serem clusterizados sobe, os *clusters* com mais de 5 passam a ser menos presentes e o processo os dispõe em grupos menores como pode ser visto no gráfico 10, onde os grupos entre 2 e 5 elementos ocupam 98.5% das situações, sendo que os outros juntos representam apenas 1,5%, valor abaixo do que a menor das representações em uma amostra com 50 elementos, onde *clusters* com mais de 11 elementos representavam 2.12% dos casos.

Clusterizando uma quantidade acima de 1000 elementos os grupos com mais de 5 já deixam de ter representatividade como pode ser visto no gráfico 11, portanto em um processo como o sugerido neste estudo, a aplicação do algoritmo hierárquico irá criar uma quantidade grande de grupos, porém com um número pequeno de elementos.

A quantidade de *clusters* gerados não é necessariamente um problema, desde que exista

Gráfico 11 – Percentual de clusters gerados com 1000 elementos.

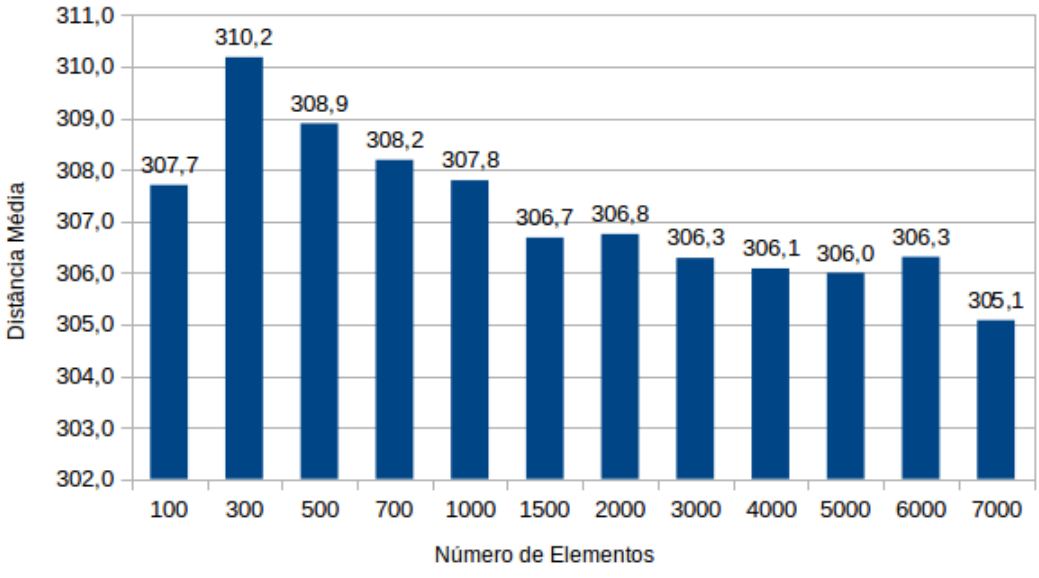


Fonte: O Autor

um balanceamento entre eles, o tamanho não é um fator primordial no estudo, porém há de se avaliar futuramente para se determinar qual seria o fator ideal que aliasse a quantidade de *clusters* e o seu tamanho, com o intuito de balancear o gerenciamento com a sua aplicabilidade, visto que um número maior de *clusters* também representa uma necessidade maior de gerenciamento, portanto há de se observar o ponto de equilíbrio onde o gerenciamento não seja sobrecarregado por uma quantidade grande de pequenos *clusters*, ou ainda, determinar o tamanho mínimo que um *cluster* deve ter para que valha a pena sua inserção no contexto.

Outro fator a se analisar é a distância entre o maior e o menor *cluster* do agrupamento, sendo este, quando o número de *clusters* é grande e inviabiliza a análise individual, um fator que determina o balanceamento dos grupos.

Gráfico 12 – Distância média de *clusters* por número de elementos.



Fonte: O Autor

O gráfico 12 demonstra a média das distâncias obtidas para clusterizações com os números determinados de elementos. Para cada número de elementos do gráfico foram executadas 100 clusterizações com a quantidade designada, para valores aleatórios de 0 a 200, e a partir delas calculada a média da distância entre o maior e o menor *cluster*.

O que foi possível perceber é que a média não tem uma variação significativa mesmo aumentando o número de elementos de 100 para 2000. A menor média registrada foi de 306.7 para 2000 elementos, enquanto a maior de 310.2 foi registrada para 300, sendo a diferença de apenas 3.5 entre elas.

Em virtude da variação de quantidade de elementos inseridos nos *clusters* assim como do a variação de tamanho destes elementos, a distância média de 306 não é um fator agravante, considerando o fato de que ainda não existe uma determinação para a melhor relação de número de elementos por *cluster*, contudo, quanto menor esta distância média, melhor o resultado do balanceamento. Assim sendo, apesar do algoritmo demonstrar uma capacidade regular de construção de um ambiente homogêneo, é perceptível que existe a possibilidade de melhoria neste quadro.

5.0.2 Assinatura de Recursos

Para que os testes pudessem ser feitos e seu resultado avaliado de alguma forma, os dados que compõe a assinatura deveriam ser provenientes de uma fonte conhecida, do contrário a assinatura gerada apenas poderia ser avaliada diretamente pelos números que a compõe, o que não seria simples devido ao grande volume de informação processada. Com os dados de um recurso conhecido, seu comportamento pode ser avaliado diretamente pela representação na assinatura, assim como alguma irregularidade poderia estar ou não presente nela.

Considerando isto, para a execução dos testes foram coletadas informações durante 200 dias de um servidor em produção, tempo suficiente para criar uma assinatura precisa sobre o comportamento do recurso, que possui uma peculiaridade em relação a outros: Este recurso possui dois núcleos de processamento, e um processo interno que roda indefinidamente e ocupa 100% de um dos núcleos, deixando apenas o outro livre para atividades da máquina. Sendo assim, a média mínima de uso do processamento deste recurso é de 50%, que corresponde a média dos dois núcleos, portanto um valor coletado abaixo deste limiar corresponde a uma falha de funcionamento no processo interno. Exatamente com esta percepção que o recurso em questão se torna interessante para os testes, visto que em um determinado dia do período coletado houve uma queda, o que deve ser levada em consideração para a construção da assinatura, e portanto não deve ser enquadrada nela.

Dentro do respectivo recurso foi instalado um processo de monitoramento, leve o suficiente para não impactar no desempenho da máquina. Este, a cada 6 segundos efetuava uma leitura sobre a quantidade de CPU e memória RAM em uso e armazenava em um arquivo de

texto, separando diariamente as leituras em arquivos específicos.

Em virtude de alguma situação eventual no funcionamento do recurso, as coletas diárias poderiam ser feitas em horários ligeiramente diferentes, ou seja, em um dia a leitura poderia executar no horário 02h34m23s e no outro dia às 02h34m25s, assim é interessante enquadrar qual o intervalo de tempo corresponde ao mesmo período. Como nos recursos podem ocorrer picos e quedas momentâneas por diversos fatores de utilização, não é interessante que estes valores sejam agrupados por um intervalo pequeno de tempo, como 5 segundos, onde por exemplo, todos as leituras efetuadas entre às 00h25:10 e 00h25:15 seriam agrupadas, isso pode gerar também na assinatura uma variação muito grande de picos, visto que em determinadas situações pode ocorrer de somente uma leitura ter sido feita no intervalo especificado.

Dentro dos testes executados, duas percepções de assinatura foram criadas, uma com uma precisão maior que pode ser melhor analisada por um sistema, e outra a ser disponibilizada por um analista, que portanto pode ter uma precisão menor, porém que proporciona visualmente um entendimento mais adequado sobre o funcionamento do recurso.

Ao enquadrar as leituras em um *TimeSlot*(TS), ou janela de tempo, como pode ser visualizado no gráfico 13, onde as leituras dentro de um mesmo intervalo de 60 segundos são consideradas como do mesmo intervalo de leituras, é perceptível que em alguns momentos o grande número de variações como as que acontecem no horário das 06h40m aproximadamente não permitem definir com precisão qual é o uso do recurso. Nesta figura, os valores que aparecem em azul são o funcionamento mínimo normal do recurso identificado pelo método, sendo que em vermelho é o máximo, portanto o funcionamento comum deste recurso se encontra entre estas duas linhas, vermelha e azul.

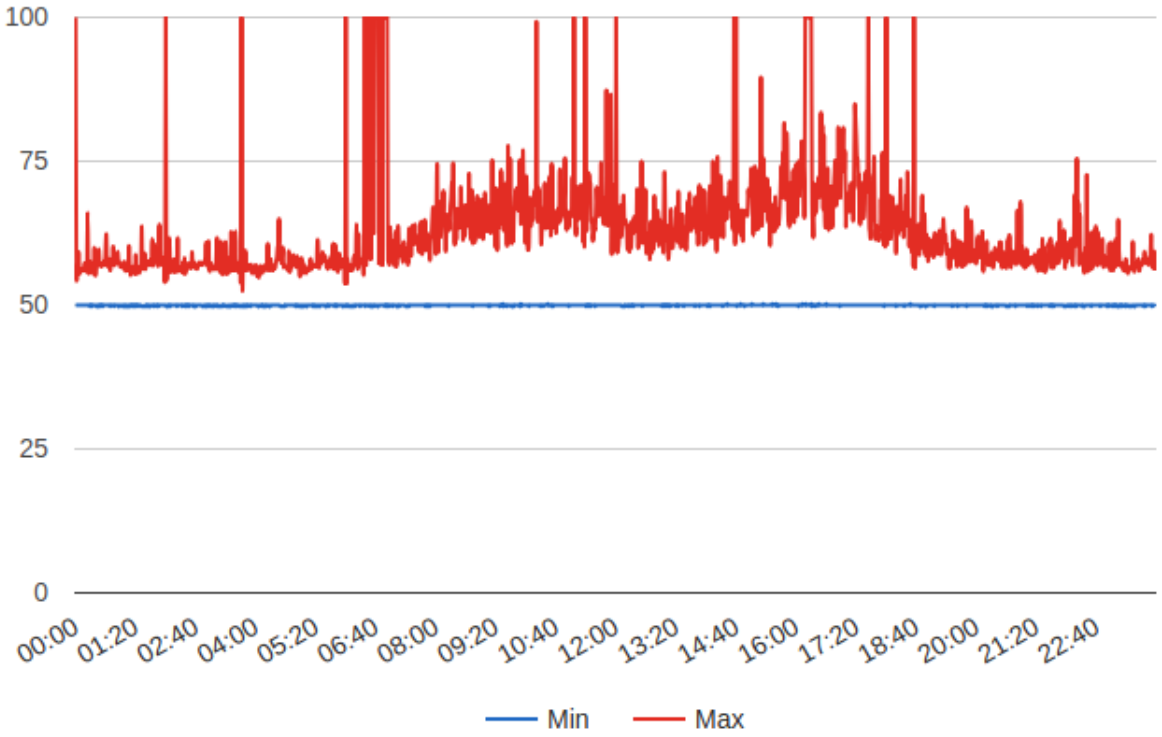
Nesta situação, até mesmo para um sistema se torna arriscado tomar uma decisão sobre a utilização do recurso, visto que esta pode ser embasada em um período de várias alterações nos próximos intervalos de tempo, como o ja citado período das 06h40m.

Assim para que estas constantes alterações sejam alinhadas, pode-se alterar o enquadramento do TS para um valor mais adequado, como acontece no gráfico 14, que passa a considerar uma janela de 100 segundos de intervalo.

Ao se alterar o período, a percepção do número de leituras na janela também é alterada, e o que pode ser visto a partir disso é uma adequação no intervalo das 06h40m, onde antes era visto uma grande sequência de picos e quedas no uso suficiente para aparecer apenas um borrão no gráfico, quando que na figura 14 este aumento é alinhado e mantido no topo, o que torna mais compreensível qual é o comportamento do recurso, assim, é possível identificar que um TS de 100 segundos é suficiente para se basear na construção da assinatura para o método com o algoritmo *K-means*.

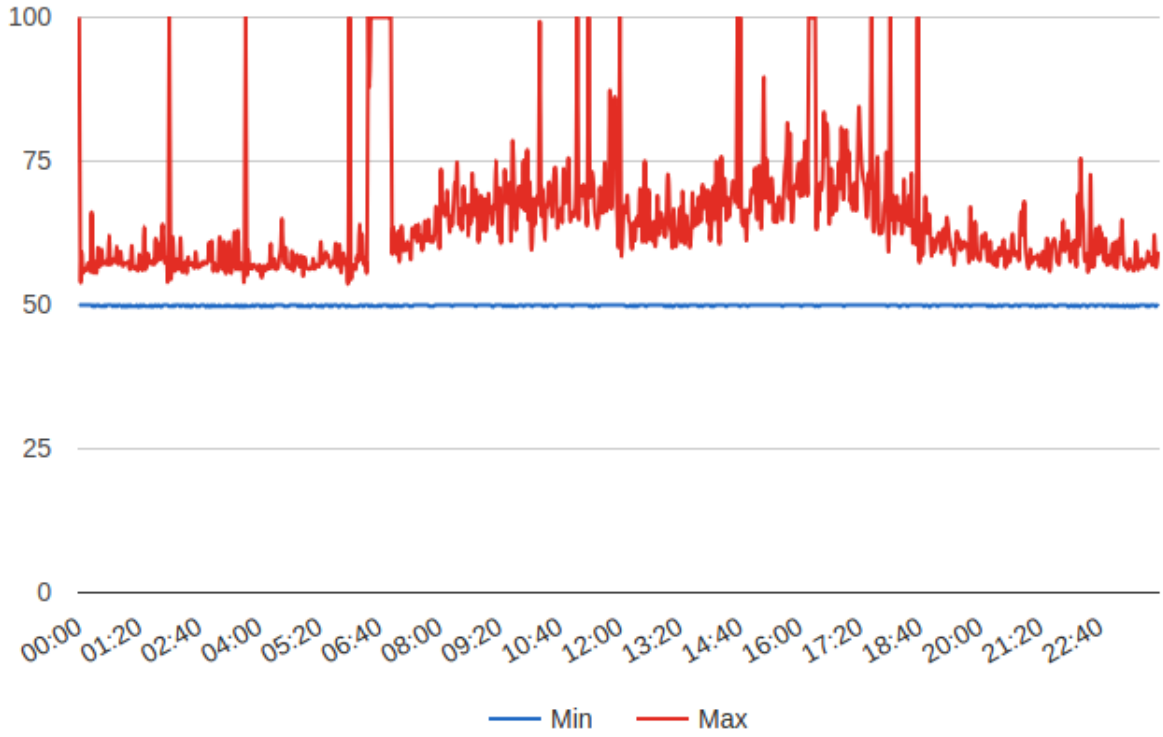
Contudo, visualmente a Assinatura Comportamental ainda não é agradável de se analisar, pois posto em um gráfico várias alternâncias de subida e descida em pontos próximos tornam difícil de determinar qual é realmente o local onde deveria estar, neste caso, a linha do limite

Gráfico 13 – Assinatura Comportamental com janela de tempo de 60 segundos.



Fonte: O Autor

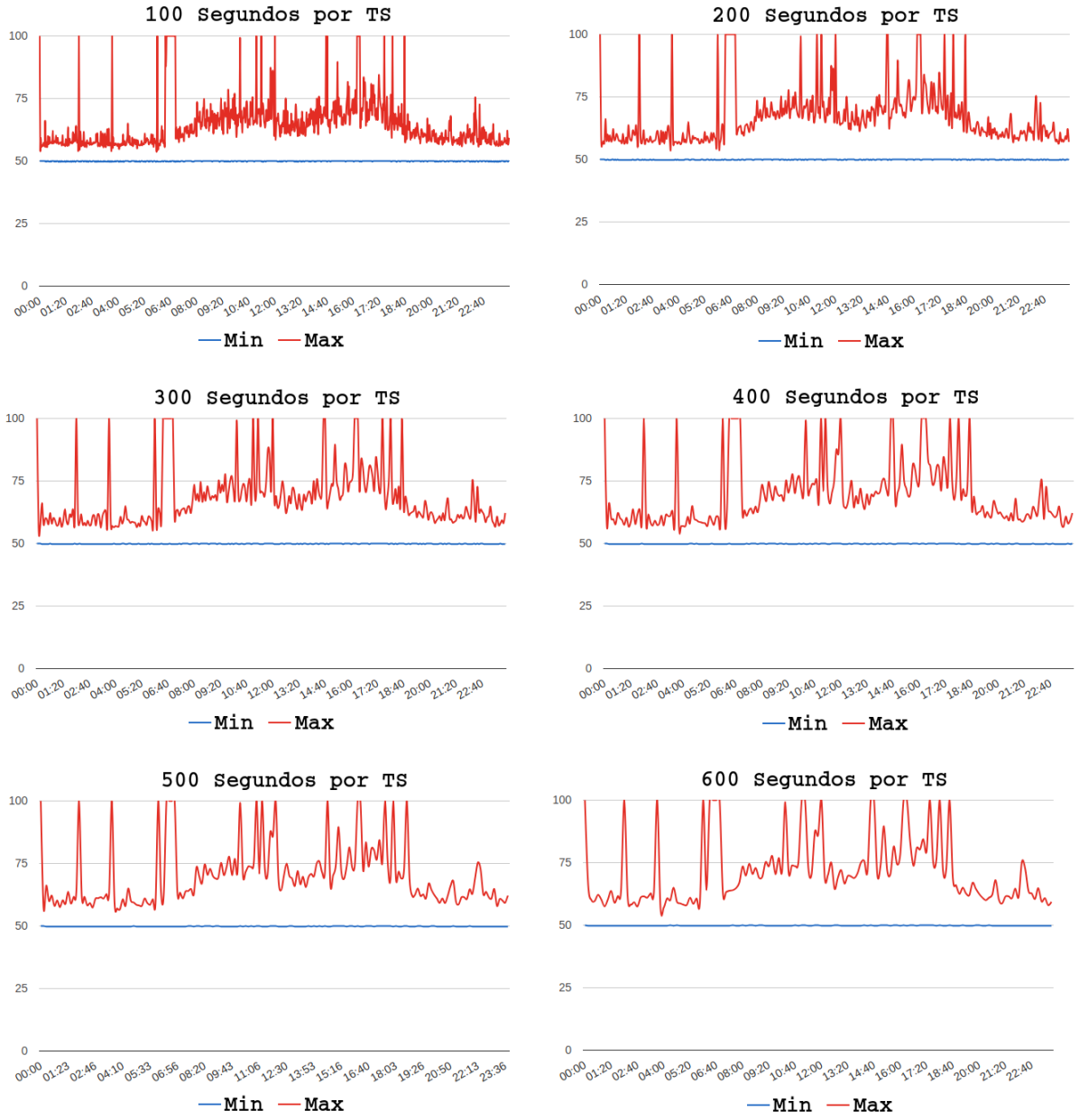
Gráfico 14 – Assinatura Comportamental com janela de tempo de 100 segundos.



Fonte: O Autor

superior, por isso testes com outras variações de TS foram executadas, e o resultado pode ser visto nos gráficos da figura 20.

Figura 20 – Assinatura Comportamental com variação de intervalo de TS.



Fonte: O Autor

A medida que aumenta o tamanho do TS, as linhas ficam mais visíveis, o que permite uma melhor averiguação para um analista. A partir de 200 segundos por TS as variações de subida e descida podem ser visualizadas com mais clareza, o que antes com 60 e 100 segundos não era possível. Subindo ainda esta janela de TS, a linha de limite superior passa a ficar bem definida com contornos mais suavizados.

Há de se atentar ao fato de que aumentar o tamanho da janela não é somente um fator visual, se este número de segundos for muito grande ele pode passar a agrupar um período muito longo do comportamento do recurso, deixando de representar no momento requisitado a sua assinatura mais precisa, por isso, após os testes ficou claro que uma janela maior que 300

segundos, apesar de visualmente ficar mais atrativa, já demonstra um risco à representação do comportamento, visto que alterações importantes podem deixar de ser expressas. Da mesma forma, um TS de 60 segundos também é muito pequeno para a representação devido a grande variação permitida, porém com 100 segundos já se percebe mais fielmente o comportamento do recurso. Fica assim estabelecido que para análise interna via um sistema que necessita de uma precisão maior, contando com variações mais confiáveis na assinatura, 100 segundos de TS é o indicado, e do outro lado, para uma análise mais visual por um especialista, uma janela de 300 segundos reflete bem o comportamento diário do recurso.

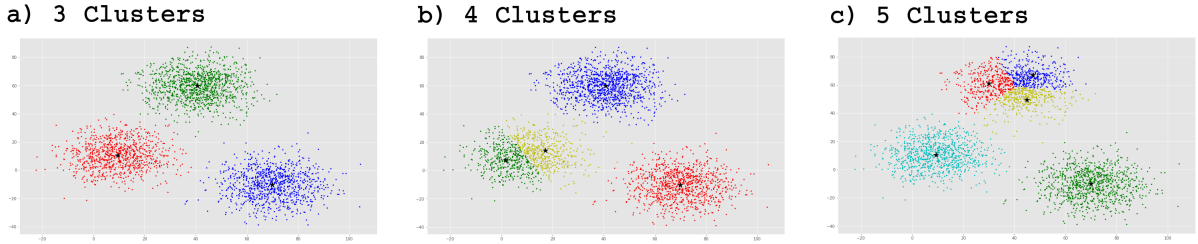
Ainda há outro fator a se analisar para as assinaturas, como o método utilizado para sua geração é o *K-means*, um número de *clusters* é necessário que seja informado, por isso para definir qual o melhor número, foram geradas assinaturas contemplando diferentes quantidades. Nos exemplos anteriores o número utilizado foi de 4 *clusters*, e como o que estava sendo avaliado eram as curvas proporcionadas pela assinatura e não a corretude dos seus limites, a quantidade utilizada tinha pouca importância, utilizando-se do bom senso e não gerando valores muito grandes como menores ou iguais a 1.

A quantidade de *clusters* escolhida funciona como um separador das leituras, como se dividisse o gráfico automaticamente em áreas de cima abaixo. Quanto menor o número de *clusters*, maior é abrangência que os grupos gerados terão sobre as leituras efetuadas, portanto elementos que poderiam não fazer parte de um determinado grupo passam a integrá-lo pois o número de *clusters* é tão pequeno que é quase solicitado para que existisse apenas um grupo no processo.

Enquanto aumenta o número de grupos, os elementos com menor proximidade ao centro do grupo que seria o principal, deixam de integrá-lo, e passam a ser considerados como leituras fora do padrão para o recurso.

Da mesma forma um número muito grande de *clusters* não é interessante, pois assim se aumenta a granularidade, criando vários grupos com um pequeno número de leituras, e separando grupos que em tese já estariam corretos. Esta situação pode ser melhor compreendida com a figura 21, onde representada a divisão de elementos por cores, a situação a com 3 *clusters* se apresenta como a que divide corretamente os elementos, e quando é aumentada para 4 ou 5 *clusters* como nas situações b e c, o algoritmo passa a partir os *clusters* para adequar a solicitação.

Figura 21 – Resultado do k-means com diferentes números de *clusters* solicitados.

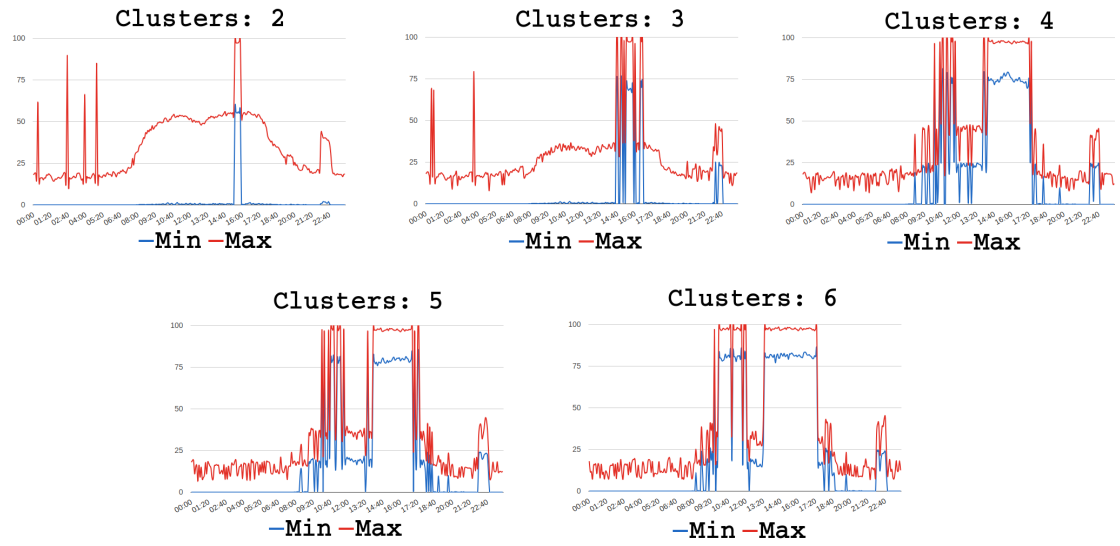


Fonte: Adaptado de (K-MEANS...,)

É seguindo esta lógica que deve-se analisar qual o melhor número de *clusters*, permitindo que sejam criados *clusters* mais próximos da perfeição, e não separando-os. A diferença na aplicação do método é que os grupos apenas consideram os valores para a janela de tempo selecionada, não utilizando valores que se encontrem ao lado em outras janelas (TS).

Para identificar a quantidade mais adequada, foram executados testes com um recurso conhecido, e neles foi buscado pelo número de *clusters* que representasse o comportamento porém sem alterações bruscas, ocasionadas por leituras esporádicas. Além disso, no referido recurso era de conhecimento a execução de rotinas pela madrugada, que por um certo momento aumentavam a utilização da CPU, portanto estas alterações deveriam aparecer no gráfico como uma ocorrência padrão do comportamento, diferentemente das alterações bruscas citadas.

Gráfico 15 – Assinaturas com 300 de TS e diferentes números de *clusters*.



Fonte: O Autor

No gráfico 15 assinaturas geradas com diferentes números de *clusters* demonstram como se comporta o método frente à estas alterações. O que pode-se perceber é que apesar do número pequeno, na assinatura com apenas 2 *clusters* o gráfico ficou menos sobrecarregado, corretamente exibindo inclusive os picos das operações que ocorrem no servidor às 01h, 03h, 05h e 6h, assim como o aumento de processamento decorrente das execuções no horário comercial.

Quando se aumenta o número de *clusters* as ocorrências das alterações voltam a ser bruscas, e deixam o gráfico sobrecarregado em várias ocasiões na assinatura. A presença de um número pequeno de *clusters* não significa que o gráfico faz a separação em áreas proporcionais para cada um deles, os *clusters* podem ter tamanhos variados devido a distância dos elementos com a centroide, criando um grupo pequeno longe de um grupo com vários elementos. Aumentar o número de *clusters* se mostrou ineficiente, já que valores deixaram de ser utilizados para composição da assinatura por terem sido repartidos em vários outros grupos.

Assim, nesta etapa os testes demonstraram que um número reduzido de *clusters* pode ser suficiente para exibição da Assinatura Comportamental, e também para análise por um sistema

uma TS de 60 segundos é suficiente, enquanto para análise visual da assinatura completa diária, 300 segundos permite uma melhor compreensão da situação do recurso em certos períodos do dia.

5.0.3 Assinatura de Clusters

Os testes de Assinatura Comportamental foram efetuados com dados obtidos de duas formas: Valores gerados randomicamente por máquinas fictícias e equipamentos reais em um período normal de funcionamento. Para que fosse testado com uma quantidade razoável de equipamentos, foram geradas assinaturas destes recursos, o que possibilitou aumentar os testes e principalmente, de início, validar as primeiras etapas da sua construção. Ter um número grande de recursos, mesmo que fictícios permitiu uma avaliação mais eficiente do método, assim como ter máquinas reais permitiu avaliar a eficácia, já que o seu funcionamento comum é conhecido.

Assim como a clusterização, as assinaturas foram criadas a partir dos recursos conectados ao sistema, fictícios ou não, sendo geradas com valores aleatórios que eram combinados e organizados em ordem temporal. Para estes recursos foi escolhida a CPU para testes, pois este costuma ser um dos componentes mais monitorados. Como o procedimento de assinatura é o mesmo para qualquer tipo de componente, não interfere que os testes possuam apenas os valores aleatórios de CPU para as máquinas fictícias, pois é o procedimento de geração de assinaturas do *cluster* que está em avaliação neste momento, e não o próprio recurso.

A construção dos valores dos recursos fictícios foi feita de forma a não permitir a todo momento uma variação extrema no decurso da assinatura, isso quer dizer que se às 09h00m os valores da assinatura estivessem em 10% de uso por exemplo, um salto para 90% não ocorreria às 09h01m, e sim um aumento mais suave. Porém situações deste tipo podem ocorrer, mas não a todo momento, assim, estes valores foram construídos através de possibilidades, onde havia 70% de chance de acontecer um salto de 10 pontos percentuais, 20% de chance para um salto de 20 pontos, e 10% para 50, sendo este salto tanto para mais quanto para menos. Isso foi feito para se aproximar do funcionamento natural de um recurso, onde existe em grande parte das situações um aumento ou declínio suave, e em outras, porém em menor proporção uma variação mais alta, assim as assinaturas não teriam grandes picos e quedas constantes.

Com esta forma, se em um determinado momento o equipamento estivesse com 40% de uso, e fosse aleatoriamente permitido um salto de até 20 pontos, um outro procedimento também de ordem aleatória possibilitaria que este salto fosse para mais ou para menos, ou seja, dos 40% atuais para 20% ou então quando o salto fosse para cima, atingindo até 60%, assim qualquer valor entre 20% e 60% poderia ser gerado. O mesmo ocorre se o salto fosse de 50 pontos, porém os 40% atuais do recurso poderiam descer até 0%, já que valores negativos e acima de 100% não são possíveis, e subir até 90%, gerando uma possível variação de valores então de 0% a 90%, permitindo inclusive a manutenção do valor em 40% se assim fosse aleatoriamente gerado.

Esta assinatura representa a leitura de um dia normal de funcionamento do recurso, por isso várias destas assinaturas são geradas para o mesmo dia, para que o procedimento possa através delas gerar a assinatura comportamental.

De início, a assinatura do *cluster* foi gerada a partir de recursos reais, portanto um grupo de 3 recursos foi utilizado para que através do conhecimento do comportamento deles fosse possível criar e avaliar a assinatura.

Nos gráficos 16 os recursos formadores abaixo da assinatura principal demonstram como as suas assinaturas contribuíram para a sua formação. Os recursos das assinaturas a e b possuem poder computacional 18, referente aos pesos dos componentes identificados para o ambiente, e a c possui poder de 56. Como a última tem um poder maior que o dobro dos outros recursos, seu formato se torna muito mais presente no resultado da assinatura principal, já que o poder computacional é o peso com que seu formato é aplicado em relação ao outros no método. Isso é bem perceptível em virtude do aumento de processamento existente na assinatura c no período entre às 8h e às 21h e que se apresenta também no resultado final, mesmo que ocorrendo de uma forma mais branda em virtude da mistura com as outras assinaturas, tornando o aumento de processamento que ocorre entre as 8h e as 10h mais suavizado.

Contudo, alguns valores das outras assinaturas de menor expressão podem ser percebidos em uma análise mais criteriosa. Enquanto as assinaturas a e c possuem limite inferior 0 por quase todo o seu período, a assinatura b tem este valor praticamente fixado em 50, o que faz com que no resultado final o limite inferior seja estabelecido sobre o valor 10.

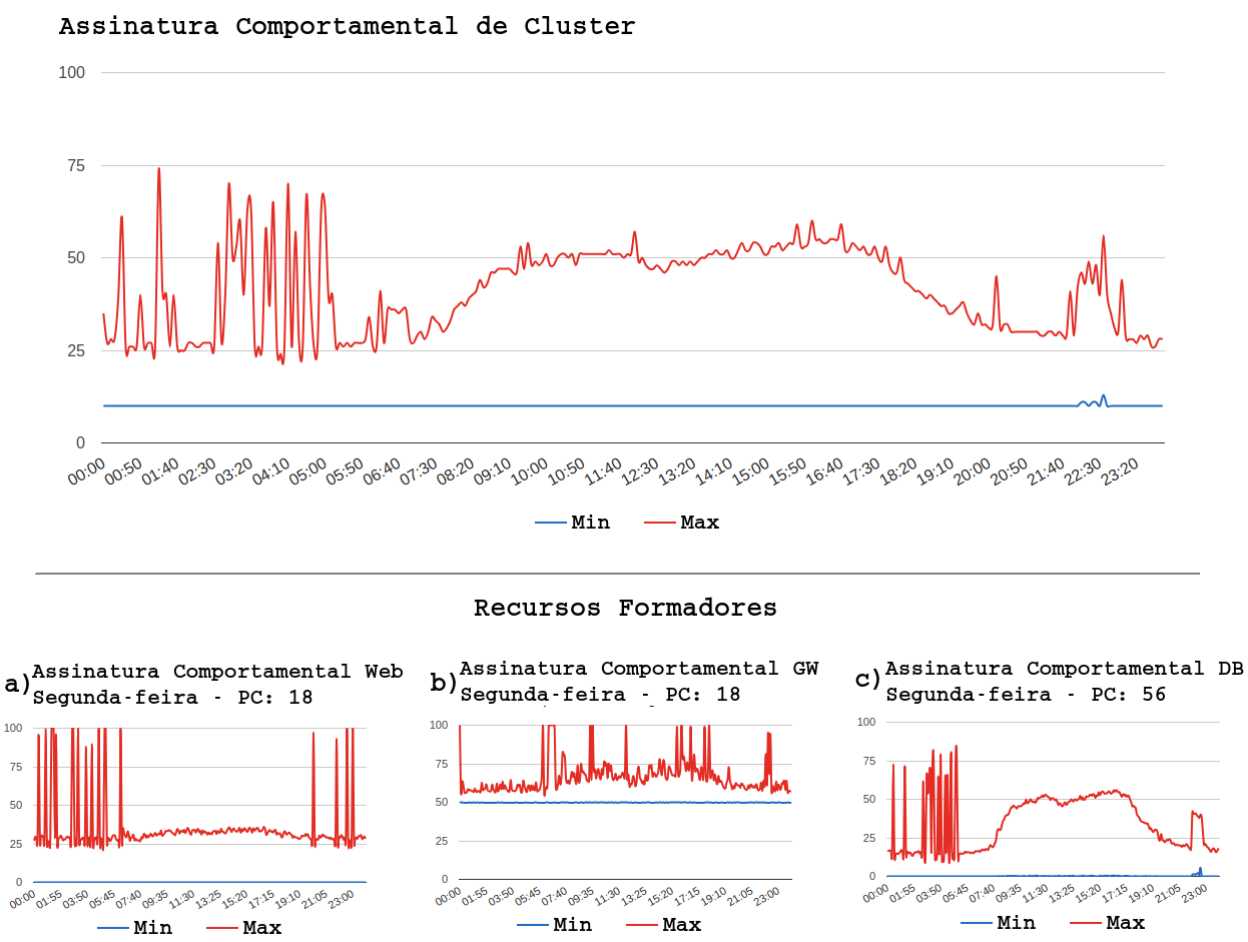
Algumas alterações no período das 8h e 21h podem ser percebidas como leves picos de uso principalmente apresentados pelo recurso da assinatura b, os quais não estão presentes nas outras durante este período, tornando a assinatura de maior peso (c) sendo expressa com leves alterações neste período.

Tendo uma assinatura com recursos reais, e com parâmetros que permitem avaliar sua veracidade possibilita que testes com recursos fictícios sejam feitos. Com eles foi possível criar um ambiente com um volume grande recursos, portanto o que se apresenta nos gráficos em 17 é uma assinatura gerada por 100 deles e, abaixo dela, 3 destes que a compõe. Na Assinatura Comportamental do *cluster* é possível perceber um limite bem definido de funcionamento, mesmo com os recursos internos apresentando assinaturas com valores tão distintos como pode ser visto nos recursos que a integram.

O padrão de funcionamento é o que pode ser bem visualizado entre as linhas de limite mínimo e máximo, portanto conhecendo-se o grupo de recursos, a arquitetura tem uma condição de escolher com uma maior capacidade qual é o *cluster* dentre os disponíveis no ambiente a receber uma solicitação de execução. Neste exemplo o *cluster* acabou possuindo um funcionamento linear, porém em certas situações estes valores podem ser distorcidos gerando picos, o que demonstraria uma menor disponibilidade para o uso.

Com estes resultados o modelo de assinatura baseado em média ponderada para os

Gráfico 16 – Assinatura do *cluster* com 3 recursos conhecidos.



Fonte: O Autor

clusters demonstrou efetividade, portanto pode-se testar em um ambiente com um número maior de elementos, mesmo que fictícios, afim de toda a arquitetura ter seu funcionamento avaliado.

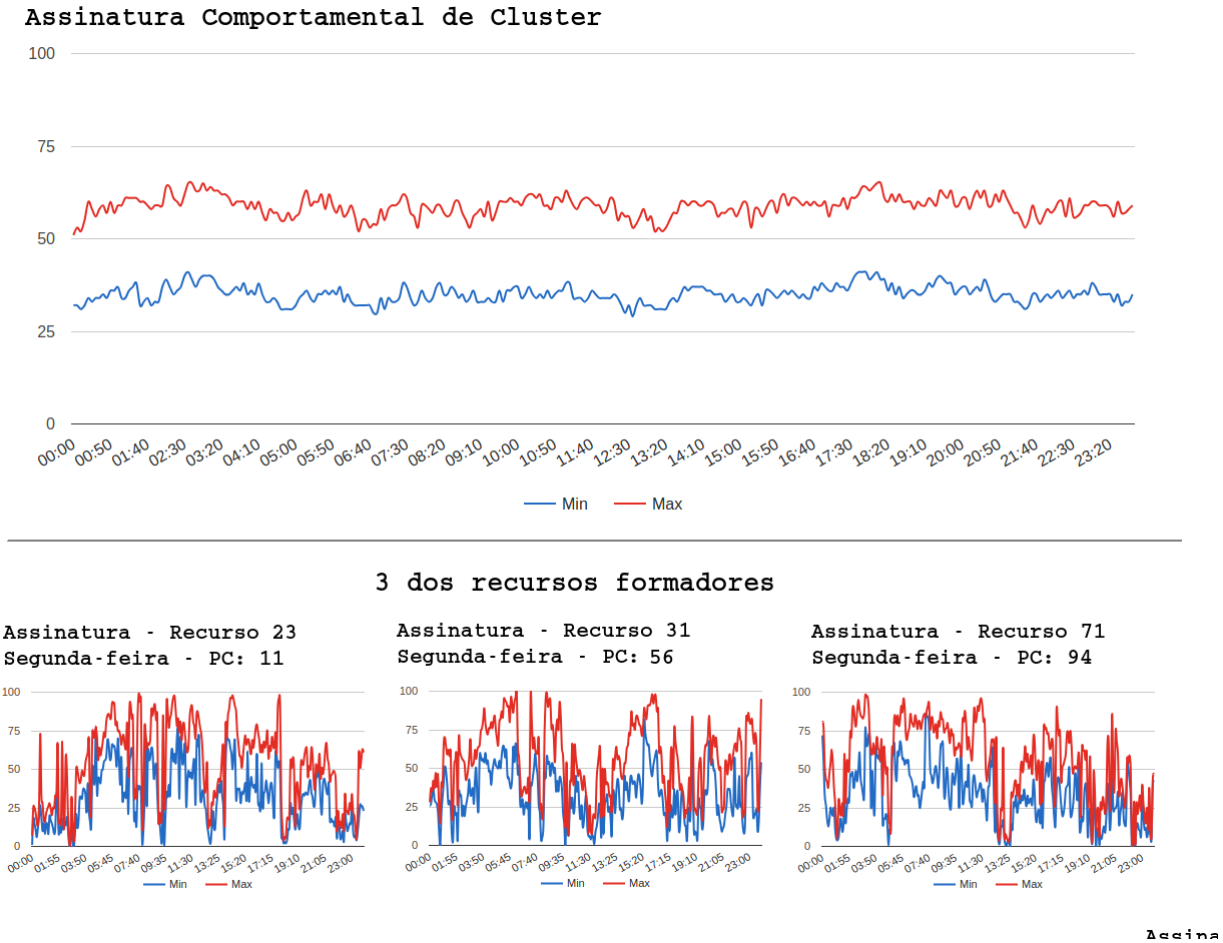
5.0.4 Arquitetura

Após terem sido testados os elementos que compõem a arquitetura, o que resta a ser avaliado é o resultado final do ambiente, o qual se apresenta como o próprio funcionamento, comunicação e principalmente a concorrência à que cada elemento é submetida.

Uma preocupação para o funcionamento da arquitetura foi o de não sobrecarregar a rede em virtude da necessidade da comunicação entre os recursos e o gerente, pois, futuramente além da própria arquitetura existiria ainda o funcionamento do escalonamento das atividades, o que gera uma comunicação extra na rede, portanto a arquitetura tem que ser leve o suficiente para não impactar no ambiente.

Para suavizar a carga, a comunicação da arquitetura foi restringida a duas etapas: a

Gráfico 17 – Assinatura do *cluster* com 100 recursos fictícios.



Fonte: O Autor

conexão, a qual envia os dados do recurso para cálculo do poder computacional e, em casos onde a assinatura já tenha sido criada, também faz o seu envio para o gerente; a outra etapa é uma comunicação diária que representa a própria assinatura comportamental do recurso, alterada em razão de dados levantados a parti de sua utilização anterior.

O arquivo com a assinatura gerada do recurso ficou em média com 4Kb de tamanho, permitindo que trafegue desapercebido no fluxo de uma rede, até pela restrição ao seu envio que limita sua comunicação em apenas uma vez durante o dia. Com isso a rede ficou livre, e mesmo com um volume grande de recursos conectados a comunicação se torna leve, permitindo que a implementação de um mecanismo de controle de escalonamento de tarefas faça um uso mais expressivo da estrutura de rede.

Um ponto muito importante na arquitetura é a percepção para um recurso conectado voluntariamente ao sistema, de que é utilizado e significativo para o ambiente como um todo, mesmo sendo este um recurso com baixo poder computacional. Esta percepção se dá pela chance que cada um tem de ser utilizado, e isso passa pela concorrência a que ele é submetida. Se um recurso com baixo poder deve concorrer com todos os recursos do sistema, possivelmente seu uso efetivo seja próximo de zero, criando uma situação de fuga ja descrita, o que não é interessante para o ambiente.

Assim como nos testes da Assinatura Comportamental do *cluster*, em virtude da necessidade de um grande número de recursos, foi utilizado do artifício de criação de recursos fictícios, bem como suas assinaturas comportamentais e poder computacional. O que ficou evidente nos testes é que, em virtude de em grande parte das ocorrências a geração de *clusters* foi de tamanho entre 2 a 5 recursos, esta concorrência se apresentou mais interessante para os recursos com menor poder computacional, visto que aqui a sua concorrência direta reduziu a um cenário com 4 outros recursos, aumentando suas chances de escolha.

6 CONCLUSÃO

Neste estudo foi elaborada uma estrutura de organização de recursos computacionais, visando proporcionar não somente um maior aproveitamento destes em momentos de subutilização, mas também um equilíbrio proporcional de uso entre recursos fortes e fracos em um ambiente distribuído. O modelo proposto utiliza um algoritmo hierárquico baseado no vizinho mais distante para agrupar recursos em *clusters* homogêneos, aumentando a chance de utilização dos recursos de menor capacidade. A distinção dos grupos para utilização em um momento de demanda é feita pela identificação de um perfil de comportamento denominado Assinatura Comportamental, a qual através de um acompanhamento traça seu padrão de utilização, permitindo identificação de momentos oportunos de uso dos grupos.

A organização em grupos dos recursos permite que aqueles de menor capacidade tenham melhores condições de concorrência pela execução de uma demanda no ambiente, isso se deve ao fato de que em um grupo a quantidade de recursos que competem por esta execução é menor em relação ao ambiente como um todo, o que favorece recursos fracos. Tendo uma maior possibilidade de participação também se mitiga a fuga de recursos do ambiente pela falta de utilização. Em conjunto, a aplicação de uma Assinatura Comportamental proporciona um parâmetro de distinção entre os grupos, além de permitir prever o comportamento e definir quem está mais apto a executar uma demanda, independente do poder computacional que este recurso possui.

Segmentos deste estudo já foram apresentados em eventos como em (SENGER; GOIS, 2016) onde foi apresentada a aplicação da abordagem em um ambiente geográfico, focando na aplicação em um sistema de grade computacional. Também em (SENGER; GOIS, 2017a) a metodologia foi descrita e aplicada dando ênfase em um ambiente distribuído, demonstrando a sequência aplicada através da utilização do algoritmo hierárquico em conjunto com o cálculo do poder computacional dos recursos. Ainda em (SENGER; GOIS, 2017b) foi demonstrada a aplicação da assinatura comportamental em *clusters* gerados pela metodologia, porém com uma abordagem diferenciada para geração da assinatura, sendo ela executada com auxílio do algoritmo *k-means*.

Assim como acontece em vários estudos, resultados muito bons foram obtidos da arquitetura, porém em certos aspectos nem tanto. Como a execução da arquitetura é modularizada, o resultado destes módulos é exposto para também se pontuar onde estão as maiores possibilidades de melhoria e, onde os fatores foram satisfatórios o suficiente.

Abordando cada etapa da execução da arquitetura e, iniciando-se pela clusterização, durante os testes ficou claro que o algoritmo escolhido para criação dos *clusters* homogêneos apresentou o que foi proposto, ou seja, uma relação de grupos com valores equilibrados. Há de se salientar que transpareceu nos testes que o balanceamento poderia ser refinado em certas ocasiões, ou seja, visivelmente existiria uma forma de melhor organizar os recursos e diminuir as

distâncias entre os grupos, o que o Algoritmo Hierárquico não permite, visto que o seu resultado é um só para a mesma relação de elementos independente de quantas vezes o processo seja executado.

Além do balanceamento, a quantidade de recursos por grupo se torna praticamente fixada em 5. Isto não é de todo um problema, contudo, quando testes sobre a razão entre o número ideal de recursos por grupo forem feitos e um valor diferente deste for apontado para alguma situação, este valor fixo deixará o sistema inflexível.

Outro fator a se avaliar é o tempo de execução da clusterização. Como o processo não ocorre a todo momento justamente para que não exista mudança frequente nos elementos do grupo, o alto tempo de clusterização não se torna um problema, mas assim como o próprio processo de clusterização, demonstra uma possibilidade de melhora.

Executados os testes, o método de clusterização baseado no Algoritmo Hierárquico demonstrou que existem fatores que podem ser melhorados e arestas que podem ser aparadas, contudo, aplicado à arquitetura proposta, ele cumpre o objetivo.

Sobre a Assinatura Comportamental, o método baseado no Algoritmo *K-means* conseguiu expressar muito bem qual o comportamento de um determinado recurso, isso ficou claro ao avaliar um recurso conhecido que passando por um momento de queda no serviço em um dia de execução não expressou na assinatura esta ocorrência. Este tipo de situação permitiria inclusive a análise para outras aplicações como detecção de anomalias, visto que a queda no serviço não se enquadraria no comportamento regular do recurso.

A assinatura também apresentou-se como um arquivo pequeno e portanto leve, o que facilita não somente o seu transporte dentro da rede, mas como é compacta a forma como se apresenta, também se torna rápida a sua análise. Parte disso se da pelo rápido processamento que o modelo permite, com isso também gerando rapidamente o arquivo da assinatura.

Com estas avaliações a Assinatura Comportamental aplicada à arquitetura se demonstrou apropriada, não apresentando necessidade de uma adequação imediata para o modelo.

A terceira etapa da arquitetura é a composição da Assinatura Comportamental do *cluster* baseado nas assinaturas dos seus recursos internos. Assim como na geração da própria assinatura dos recursos, o tempo para sua criação também foi baixo, o que evita um desperdício de processamento pelo equipamento responsável pela gerência. De imediato isso parece irrelevante, porém considerando que o método de clusterização gera um número grande de *clusters* também são muitas as assinaturas para serem processadas, por isso um baixo tempo de processamento é imprescindível.

Como a expressão da Assinatura Comportamental do *cluster* é baseada em uma média ponderada, o impacto que cada recurso tem no resultado final é apropriada, visto que cada um contribui na mesma medida que o seu poder computacional para o poder total do *cluster*. Desta maneira, para a proposta da arquitetura o modelo apresentado para geração desta assinatura é adequado, não apresentando nos testes executados nenhum fator que necessite de um

aprimoramento.

Analisando a arquitetura em um contexto geral, o resultado foi satisfatório. Para os objetivos traçados, mesmo pontos que apresentaram a possibilidade de serem aplicadas melhorias, como a clusterização por Algoritmo Hierárquico, atingem o que se foi requerido, fazendo com que estas possíveis melhorias acrescentem em qualidade.

O resgate dos recursos de baixo potencial para o ambiente contribui para mitigação da fuga de recursos, contudo, o quanto a arquitetura implicará na efetiva utilização destes, somente será possível quantificar a partir de quando o modelo for colocado em prática com modelos de escalonamento de tarefas, o que ainda demandará muitos estudos a partir deste.

6.0.1 Trabalhos Futuros

O balanceamento dos *clusters* assim como o tempo de clusterização não foram os mais satisfatórios, por isso a utilização de outro algoritmo como o K-Balance que tem bons resultados neste segmento pode ser uma saída para melhorar estes itens apontados.

Além disso, uma análise automática das demandas do ambiente relativa ao componente mais requisitado dos recursos (RAM, CPU, HD), e assim ponderar a esse um peso maior, permitindo que os cálculos priorizem aquele que é o componente mais importante para o ambiente, seria um avanço interessante para melhor aproveitar os recursos integrados.

Para efetivar este desenvolvimento, um estudo sobre o escalonamento das tarefas em um ambiente utilizando recursos em *clusters*, o modo de escolha de cada recurso e como implantar o desenvolvimento das tarefas também seriam de grande valia para avaliar na prática quanto um recurso de baixo poder computacional receberia a mais em tarefas para execução comparado a um ambiente onde houvesse a necessidade de uma concorrência direta a outros recursos.

Por fim, a designação de qual a melhor proporção de recursos por *clusters* também é necessária, pois um grande número de *clusters* gerado também impacta de forma negativa na sua gerência, enquanto um pequeno número prejudica a concorrência de recursos fracos, por isso é indicado encontrar a proporção ideal entre eles.

REFERÊNCIAS

ANDERBERG, M. R. **Cluster analysis for applications**. Academic Press, 1973. (Probability and mathematical statistics). ISBN 9780120576500. Disponível em: <https://books.google.com.br/books?id=BZBYAAAAMAAJ>.

BAUM, L. E.; PETRIE, T. Statistical inference for probabilistic functions of finite state markov chains. **The Annals of Mathematical Statistics**, The Institute of Mathematical Statistics, v. 37, p. 1554–1563, 1966. Disponível em: <https://projecteuclid.org/euclid.aoms/1177699147>.

BLUM, C. Ant colony optimization: Introduction and recent trends. **Physics of Life Reviews**, Elsevier, p. 353–373, 2005. ISSN 1571-0645. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1571064505000333>.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. **Swarm Intelligence: From Natural to Artificial Systems**. New York, USA: Oxford University Press, 1999. 25–31 p.

BRADLEY, P. S.; FAYYAD, U. M. **Refining Initial Points for K-Means Clustering**. In: . [S.l.]: Morgan kaufmann, 1998. p. 91–99.

CARVALHO, L.F. et al. **Digital signature of network segment for healthcare environments support**. p. 299–309, dez. 2014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1959031814001067>.

COLLINS, T. et al. **World E-Waste Map Reveals National Volumes, International Flows**. dez. 2013. Disponível em: <https://i.unu.edu/media/unu.edu/news/41225/World-E-Waste-Map-Reveals-National-Volumes-International-Flows.pdf>.

DAVIDSON, I.; RAVI, S.S. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. **Knowledge Discovery in Databases: PKDD 2005**, Springer US, p. 59–70, 2005. Disponível em: <http://link.springer.com/chapter/10.1007/11564126_11>.

____. Using instance-level constraints in agglomerative hierarchical clustering theoretical and empirical results. **Data Mining and Knowledge Discovery**, Springer US, 2008. Disponível em: <http://link.springer.com/article/10.1007/s10618-008-0103-4/>.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization: Artificial ants as a computational intelligence technique. **IEEE Computional Intelligence Magazine**, IEEE, 2007. ISSN 1556-603X. Disponível em: <http://ieeexplore.ieee.org/document/4129846/>.

DZUBECK, F. The next killer app: A grid. **Network World**, p. 35, 2002. Disponível em: <https://books.google.com.br/books?id=yBgEAAAAMBAJ&pg=PT1&lpg=PT1&dq=the+next+killer+app:+the+grid&source=bl&ots=UDzyu27Am3&sig=pAluVFeHtz5jcgzorTjj2ORCuHU&hl=pt-BR&sa=X&ved=0ahUKEwjl84SkzMzQAhUFfZAKHVoxBTgQ6AEITzAG#v=onepage&q=dzubeck&f=false>.

ENDO, Y.; HAMASUNA, Y.; MIYAMOTO, S. Agglomerative hierarchical clustering for data with tolerance. **IEEE International Conference on Granular Computing**, IEEE, Nov 2007. Disponível em: <http://ieeexplore.ieee.org/document/4403132/>.

FANG, C.; JIN, W.; MA, J. k-means algorithms for clustering analysis with frequency sensitive discrepancy metrics. **Proc. 5th Berkeley Symposium on Mathemtical Statistics and Probability**, p. 281–297, 1967.

FARELLY, R. **K Means Clustering**. 2012. Disponível em: <http://rossfarrelly.blogspot.com.br/2012/12/k-means-clustering.html>.

FERREIRA, L. et al. **Introduction to Grid Computing with Globus**. 2nd. ed. Cambridge, MA, USA: Redbooks, 2003.

FORGY, E. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. **Biometrics**, v. 21, n. 3, p. 768–769, 1965.

FOSTER, I. **The grid: blueprint for a new computing infrastructure**. [S.l.]: Morgan Kaufmann Publishers Inc, 1988. ISBN 1-55860-475-8.

____. **What is the Grid? A Three Point Checklist**. GRID Today, jul. 2002.

FOSTER, I. et al. **The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets**. Jul 2000. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804500901103>.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. **The Anatomy of the Grid**. 2001. Disponível em: <https://www.globus.org/sites/default/files/anatomy.pdf>.

GóIS, L. Ap. de. **Estratégias Para Comercialização de Recursos Computacionais em Desktop Grids**. Tese (Doutorado) — Universidade Estadual de Campinas, 2009.

HAIDER, S.; NAZIR, B. Dynamic and adaptive fault tolerant scheduling with qos consideration in computational grid. **Special Section on Emerging Trends, issues, and challenges in energy-efficient cloud compution**, IEEE Access, v. 5, 2017. Disponível em: <http://ieeexplore.ieee.org/document/7783218/>.

HANDL, J.; KNOWLES, J.; DORIGO, M. **Ant-based clustering: a comparative study of its relative performance with respect to -means, average link and 1d-som**. CiteSeer, 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.1275>.

HANMIN, Y.; HAO, L.; QIANTING, S. An improved semi-supervised k-means clustering algorithm. **Procedia Engineering**, Elsevier, p. 4325–4329, 2012. ISSN 1877-7058. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877705812006753>.

HAUBERG, S. et al. Scalable robust principal component analysis using grassmann averages. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 38, 2016. ISSN 0162-8828. Disponível em: <http://ieeexplore.ieee.org/document/7364267/>.

HAYKIN, S. **Neural networks: a comprehensive foundation**. Hamilton, Ontario, Canada: Bookman, 2008. 36 p.

JIANG, F.; SUI, Y.; CAO, C. An information entropy-based approach to outlier detection in rough sets. **Expert Systems with Applications**, v. 37, p. 6338–6344, September 2010.

JIANG, H.; YU, Q.; GONG, Y. An improved ant colony clustering algorithm. **3rd International Conference on Biomedical Engineering and Informatics (BMEI**, Elsevier, 2010.

JOLLIFFE, I.T. **Principal Component Analysis**. [S.l.: s.n.], 2002. v. 2. 518 p.

K-MEANS Clustering in Python. <https://mubaris.com/2017/10/01/kmeans-clustering-in-python/>. Accessed: 2018-03-11.

KAMARUNISHA, M.; RANICHANDRA, S.; RAJAGOPAL, T.K.P. Recitation of load balancing algorithms in grid computing environment using policies and strategies - an approach. **International Journal of Scientific & Engineering Research**, v. 2, p. 7, Mar 2011. ISSN 2229-5518.

LI, J. et al. Study on robot path collision avoidance planning based on the improved ant colony algorithm. **8th International Conference on Intelligent Human-Machine Systems and Cybernetics**, IEEE, 2016. Disponível em: <http://ieeexplore.ieee.org/document/7783897/>.

LIU, Z. et al. An integrated method for anomaly detection from massive system logs. **IEEE Access**, p. 1–1, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8371223/authors>.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. **Proc. 5th Berkeley Symposium on Mathemtical Statistics and Probability**, p. 281–297, 1967.

MARIN, J. M. M.; CAMARA, S. B.; FUENTES, J. M. What does grid information technology really mean? definitions, taxonomy and implications in the organisational field. **Technology Analysis & Strategic Management**, v. 21, p. 491–513, 2009.

MASSIN, R.; MARTRET, C.J.Le; CIBLAT, P. Distributed clustering algorithms in group-based ad hoc networks. **European Signal Processing Conference**, Mar 2015.

NOROUZI, M.; AKBARPOUR, S. **Data Processing in Grid Systems by Using Cluster Algorithms**. maio 2013. Disponível em: <http://ieeexplore.ieee.org/document/6615312/>.

NORUSIS, M. J. **IBM SPSS Statistics 19 Statistical Procedures Companion**. 1st. ed. [S.l.]: Addison Wesley, 2011. ISBN 0-321-74842-5.

OHNO, Y. et al. Anomaly detection system using resource pattern learning. **Software Technologies for Future Dependable Distributed Systems**, IEEE, v. 5, 2009. Disponível em: <http://ieeexplore.ieee.org/document/7975846/>.

PROENçA, M. L. Jr. et al. **Digital signature to help network management using flow analysis**. p. 299–309, maio 2015. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1959031814001067>.

SCHOPF, J. M. **Grids: The Top Ten Questions**. 5th International Symposium on High Performance Distributed Computing, p. 103–111, 2002.

SEHATBAKHSH, N. et al. Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devices. In: **2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)**. [S.l.: s.n.], 2018. p. 1–8.

SENGER, W.; GOIS, L. Ap. de. **Alocação de recursos geograficamente distribuídos em clusters homogêneos.** In: . UTFPR, 2016. p. 55–58. ISSN 2526-1371. Disponível em: <http://www.wpccg.pro.br/volumes/volume001/proceedings/wpccg-2016>.

____. Homogeneous clusters allocation of computational resources. **2017 12th Iberian Conference on Information Systems and Technologies (CISTI)**, IEEE, v. 5, 2017. Disponível em: <http://ieeexplore.ieee.org/document/7975846/>.

____. **Assinatura Comportamental e Detecção de Anomalias de Recursos Computacionais Utilizando K-means**. 2017.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. 1st. ed. Boston, MA, USA: Addison Wesley, 2005. ISBN 0321321367.

TAYLOR, N. J. The impact of public grid computing portolio composition on adoption intentions. **The Journal of Computer Information Systems**, Oct 2006.

YILDIRIM, N.; UZUNOGLU, B. Association rules for clustering algorithms for data mining of temporal power ramp balance. **International Conference on Cyberworlds**, IEEE, Oct 2015. Disponível em: <http://ieeexplore.ieee.org/document/7398419/?reload=true&arnumber=7398419>.