Optimal Cloud Resource Allocation by Means of the Analytic Hierarchy Process

Valter Messias, Julio Estrella, Ricardo Ehlers
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brasil
{valterrm, jcezar, ehlers}@icmc.usp.br

***Abstract*—Improve resource provisioning for applications hosted in the cloud is a major research challenge. This is because is necessary to address two conflicting objectives: meet customer requirements; and save money. As there is a delay, which can take minutes, between the request for a new resource and it be ready for use, it is necessary to predict the future demand for each time interval. However, there is not a prediction model that is appropriate in all cases. To resolve this problem, it is proposed in this paper, the combination of different forecasting models by the analytic hierarchy process. In this way, we intend to create a generic solution, able to optimize the allocation of resources to several applications types, with different demand types. The results obtained by simulation show that our proposal achieves this goal.**

***Keywords*—cloud computing, multicriteria optimization, analytic hierarchy process.**

I. Introdução

Com a computação em nuvem surge um novo paradigma, no qual os recursos computacionais passam a ser oferecidos como serviços e o usuário paga somente pela quantidade e tempo em que usar esses recursos. Uma das principais características deste novo paradigma é a elasticidade, que permite alocar e desalocar grandes quantidades de recursos em tempo de execução. Para o usuário, os recursos parecem ser ilimitados, e podem ser adquiridos em qualquer quantidade e a qualquer tempo [1].

Para tirar vantagem dessa nova realidade, as empresas, cada vez mais, estão migrando suas aplicações Web para a nuvem [2]. Os objetivos desejados com o uso da infraestrutura da nuvem são praticamente dois: evitar o não cumprimento dos acordos de níveis de serviço (SLA - Service Level Agreements), em caso de aumento de demanda; e reduzir os custos com recursos ociosos, em caso de queda na demanda. Encontrar um ponto de equilibrio entre esses dois objetivos é um grande desafio.

Provedores de infraestrutura, como a Amazon, oferecem a possibilidade do usuário alocar recursos pagando um preço fixo por cada hora em que usar esse recurso. Desse modo, empresas que usam essa infraestrutura precisam decidir a quantidade de recursos necessária a cada intervalo de tempo.

A ferramenta utilizada para decidir quando e como alocar os recursos é chamada de autoscaling e pode ser divídida em dois tipos: reativas e proativas. As técnicas reativas monitoram os eventos do sistema (uso da CPU, uso da memória, tamanho da fila, etc.) e decidem por alocar ou desalocar recursos quando esses eventos excedem um certo valor limite (*threshold*). Por outro lado, as técnicas proativas tentam prever a quantidade de recursos necessária no próximo período de tempo, para se antecipar a eventos indesejados.

O problema das técnicas reativas é que, frequentemente, o tempo para reagir é insuficiente. Isso ocorre porque o tempo para um recurso na nuvem (nesse caso, uma máquina virtual) estar pronto para uso após ter sido solicitado pode levar minutos [3], tempo suficiente para que o sistema fique sobrecarregado. No caso das técnicas proativas, existem vários modelos de previsão de séries temporais que podem ser utilizados para prever a demanda. No entanto, não existe um modelo que seja adequado para todos os tipos de séries temporais, geradas por diferentes tipos de aplicações. Além disso, uma mesma aplicação pode apresentar mudanças nas carecterísticas de sua demanda ao longo do tempo, fazendo com que um modelo que estava prevendo bem passe a não ser mais adequado.

Neste trabalho, é proposto um autoscaling proativo, baseado na combinação de modelos de previsão de séries temporais por meio do método de análise hierárquica (AHP) [4]. O AHP é um método para auxiliar a tomada de decisões, quando essas envolvem mais de um objetivo. No nosso caso, os objetivos são: minimizar a quantidade de recursos subprovisionados e, consequentemente, o número de requisições não atendidadas; e minimizar o número de recursos superprovisionados, ou ociosos, evitando gastos desnecessários. Nosso sistema foi modelado como uma fila $M/M/m$. Para realizar os experimentos foram utilizados três logs (dados com informações sobre as requisições recebidas) extraídos de servidores Web reais, com diferentes características.

O restante deste trabalho está organizado como segue: na próxima seção serão apresentados os trabalhos relacionados. Na seção III, o referencial teórico será apresentado, incluindo uma explanação sobre o método AHP e sobre séries temporais. A seção IV descreve a nossa proposta de solução para o problema. Em seguida, na seção V, a avaliação da proposta será apresentada e os resultados serão discutidos. Finalmente, na seção VI, será apresentada a conclusão e propostas de trabalhos futuros.

II. Trabalhos Relacionados

Nesta seção iremos apresentar uma relação de trabalhos, relevantes, relacionados a previsão de demanda e alocação de recursos no ambiente de computação em nuvem.

978-1-4673-9143-6/15/$31.00 ©2015 IEEE

Técnicas reativas baseadas em *threshold* são muito populares em provedores de infraestrutura na nuvem, como a Amazon EC2, e ferramentas distribuídas por terceiros, como o RightScale [5]. Essa ferramenta permite definir regras de desempenho, tais como limites superiores e inferiores, para determinada variável (por exemplo, 30% e 70% de carga na CPU). Essa técnica requer um esforço extra por parte do usuário, que precisa selecionar qual variável será considerada, além de ter que definir vários parâmetros [3]. Alguns trabalhos, como o de Belaglazov et al. [6] e o de Lim et al. [7] usam técnicas adaptativas para definir o *threshold* dinamicamente.

Outros trabalhos propoem técnicas reativas mais sofisticadas, baseadas na teoria de controle. Ali-Eldin et al. [8] propõem combinar dois tipos de controles, um proativo para *scaling down* (diminuir a quantidade de recursos) e um reativo para *scaling up* (aumentar a quantidade de recursos). Padala et al. [9] apresentaram um controle adaptativo do tipo MIMO (Multiple-input multiple-output) que usa um modelo ARMA de segunda ordem (Auto Regressive Moving Average) para modelar a relação entre a quantidade de recursos e o desempenho do sistema. O trabalho proposto por Kalyvianaki et al. [10] usa um controle SISO (Single-input single-output) e um MIMO para determinar a alocação de recursos de acordo com a carga da CPU do sistema, por meio de um filtro de Kalman. Nos trabalhos de Wang et al. [11] e Xu et al. [12] é aplicado um controle *fuzzy* adaptativo para estimar a carga da CPU de acordo com a carga de trabalho do sistema. O problema das técnicas baseadas em teoria de controle é a dificuldade de encontrar parâmetros adequados para o controle, o que é feito por tentativa e erro.

Finalmente, entre os trabalhos que usam técnicas proativas para previsão da demanda e alocação de recursos na nuvem, Jiang et al. [13] propõem o uso de regressão linear para prever a demanda e, em seguida, o número de recursos é calculado usando um modelo de fila $M/M/m$. Roy et al. [14] usa um modelo ARMA de segunda ordem para prever a demanda. Os experimentos foram realizados usando o *log* Fifa World Cup 98 [15]. Após a previsão, uma função de otimização foi usada para determinar o número de recursos a ser alocado.

Fernandez et al. [16] apresentam um autoscaling que suporta infraestruturas de nuvem heterogêneas. Uma árvore de decisão foi usada para determinar a melhor combinação de recursos entre as diferentes infraestruturas. Herbst et al. [17] propõem um classificador para selecionar dinamicamente o melhor método de previsão para determinado tipo de série. Baseada nos objetivos dos usuários, uma árvore de decisão define o melhor método de previsão por meio de mecanismos de *feedback* que avaliam e comparam a acurácia de diferentes métodos de previsão. Balaji et al. [18] apresentaram um estudo comparativo sobre o desempenho de dois modelos de previsão (Holt-Winters e ARIMA) usando uma carga de trabalho extraída dos servidores Web da NASA [19].

A lacuna deixada pelos trabalhos apresentados nessa seção é o não tratamento do problema de maneira multiobjetiva, o que é essencial para otimizar os resultados. Outro ponto é não considerar a combinação de modelos de previsão no processo de alocação de recursos.

III. Referencial teórico

A. Séries temporais

Uma série temporal é uma sequencia de pontos ordenados, observados em intervalos uniformes de tempo. Uma série temporal é frequentemente representada por uma função matemática que determina sua geração. Seus componentes podem ser determinísticos ou estocásticos. São determinísticos quando podem ser representados por uma função matemática, e estocásticos caso contrário.

O propósito de um modelo de previsão de séries temporais é: separar os componentes determinísticos e aleatórios da série; encontrar uma função matemática que represente os componentes determinísticos; e estimar os componentes aleatórios usando regras de probabilidade. A seguir serão descritos os principais modelos de previsão de séries temporais.

Principais modelos de previsão

NAIVE: O modelo NAIVE, ou ingênuo, é muito simples. Este modelo assume que a próxima observação será igual a anterior. Necessita de apenas um ponto (último valor observado) para funcionar.

Auto-Regressivo (AR): Neste modelo, a previsão do próximo ponto da série é calculada baseando-se em uma combinação linear das previsões anteriores [20]. O usuário precisa informar um parâmetro (p) que define quantas previsões passadas devem ser consideradas pelo modelo.

Médias Móveis (MA): O modelo de médias móveis é similar ao modelo auto-regressivo. A diferença é que ao invés de usar as previsões passadas, os erros de previsão são usados como base para um modelo de combinação linear [20]. Esse modelo possui um parametro (q), que determina quantos erros passados devem ser considerados no modelo.

Auto-Regressivo de Médias Móveis (ARMA): O modelo ARMA é uma combinação dos modelos AR e MA. Este modelo possui dois parâmetros. O primeiro (p) define a ordem do modelo auto-regressivo e o segundo (q) a ordem do modelo de médias móveis.

Auto-Regressivo Integrado com Médias Móveis (ARIMA): O modelo ARIMA, proposto em [21], é definido por seis parâmetros (*p*, *d*, *q*) e (*P*, *D*, *Q*). Os primeiros três definem um modelo para os componentes de ruído e tendência da série. Os três últimos são opcionais, e definem um modelo para o componente de sazonalidade. *P* ou *p* definem a ordem do modelo auto-regressivo, *D* ou *d* definem a ordem da integração (necessária quando a série não é estacionária), e *Q* ou *q* definem a ordem do modelo de médias móveis [17]. Em [22] é proposta uma função para estimar de maneira automática esses parâmetros.

Alisamento Exponencial Simples (SES): Alisamento exponencial é uma técnica que atribui pesos exponencialmente decrescentes ao longo do tempo para as previsões passadas. Desse modo, como os maiores pesos são atribuidos as previsões mais recentes, este modelo é mais sensivel a capturar tendências em uma série temporal.

Alisamento Exponencial de Holt-Winters: Este modelo é uma extensão do modelo SES para capturar sazonalidade. Seu trabalho é dividir a série em três componentes: nível, tendência e sazonalidade. Para cada um dos componentes separados é, então, aplicado o SES. Em seguida, a série é reconstruida. Há duas variações para este modelo, aditivo e multiplicativo, que diferem de acordo com o padrão do componente sazonal [20].

Alisamento Exponencial Estendido (ETS): O modelo ETS, proposto em [22], tem por objetivo escolher, de acordo com a série, o melhor modelo entre o SES, Holt-Winters aditivo ou multiplicativo.

B. Método de análise hierárquica (AHP)

O método AHP (*Analytic Hierarchy Process*), proposto por Tomas L. Saaty no início da década de 70, é um método de apoio a decisão multicritério muito utilizado na resolução de problemas que envolvem mais de um objetivo.

O AHP busca tratar a complexidade de um problema dividindo-o em fatores e níveis, e estabelecendo relações de prioridade entre esses fatores. Para a aplicação dessa metodologia é necessário estruturar os critérios (considerados no processo de decisão) e as alternativas (objeto da decisão) de forma hierárquica, sendo que no primeiro nível da hierarquia está o propósito geral do problema, no segundo os critérios e no terceiro as alternativas.

Para exemplificar esse processo, vamos considerar que uma determinada família deseja comprar um carro novo. Cada membro da família tem preferência por determinado modelo, de acordo com o critério que mais lhes agradam. O pai prefere o modelo A, porque é mais econômico. A mãe prefere o modelo B, porque é mais confortável. Já o filho adolescente prefere o modelo C, porque é mais estiloso. Dessa maneira, a família precisa tomar uma decisão que envolve multíplos critérios e alternativas. A Figura 1 ilustra o problema de maneira hierárquica, como determina o AHP.

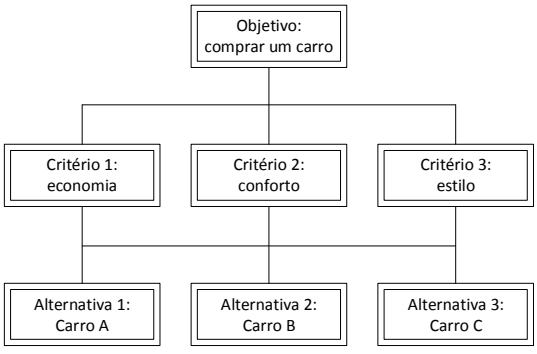

Figura 1. Problema modelado de maneira hierárquica.

Após a definição da hierarquia, o próximo passo é definir as relações de prioridade entre as alternativas, dentro de cada critério, e entre os critérios. Para estabelecer essas relações é necessário comparar par a par as alternativas, dentro do contexto de determinado critério, e definir a importância relativa de uma sobre a outra. Em [4] é proposto uma tabela de escalas para ser usada na definição das relações de prioridade. A Tabela I ilustra as escalas propostas.

Tabela I. ESCALAS DE COMPARAÇÃO SEGUNDO SAATY

Escala	Descrição
1	Elementos tem a mesma importância entre si
3	Um dos elementos tem moderada importância em relação a outro
5	Um dos elementos tem forte importância em relação a outro
7	Um dos elementos tem importância muito forte em relação a outro
9	Um dos elementos tem extrema importância em relação a outro

Incrementos de 0.1 são permitidos entre os valores propostos na Tabela I, quando se desejar ajustes mais finos.

Baseado nas escalas propostas na Tabela I, são criadas matrizes de julgamentos para as alternativas, dentro de cada critério, e para os critérios. No nosso exemplo, serão criadas quatro matrizes de julgamentos: uma para definir a relação de importância entre as alternativas dentro do critério "economia"; uma para definir a relação de importância entre as alternativas dentro do critério "conforto"; outra para definir a relação de importância entre as alternativas dentro do critério "estilo"; e outra para definir a relação de importância entre os critérios. As Tabelas II e III ilustram essas matrizes de julgamento.

Tabela II. MATRIZES DE JULGAMENTO DAS ALTERNATIVAS

Economia	Carro A	Carro B	Carro C	PML
Carro A	1	3	7	0.5889
Carro B	1/3	1	5	0.3391
Carro C	1/7	1/5	1	0.0720
Conforto	**Carro A**	**Carro B**	**Carro C**	**PML**
Carro A	1	1/5	1/2	0.1211
Carro B	5	1	3	0.6414
Carro C	2	1/3	1	0.2375
Estilo	**Carro A**	**Carro B**	**Carro C**	**PML**
Carro A	1	2	1/5	0.1615
Carro B	1/2	1	1/9	0.0813
Carro C	5	9	1	0.7572

Tabela III. MATRIZ DE JULGAMENTO DOS CRITÉRIOS

Critério	Economia	Conforto	Estilo	VPC
Economia	1	3	5	0.6414
Conforto	1/3	1	2	0.2375
Estilo	1/5	1/2	1	0.1211

Na Tabela II, podemos notar que o Carro A tem importância moderadamente superior (3) ao Carro B e muito fortemente superior (7) ao Carro C no critério "economia". Desse modo, a importância do Carro B e do Carro C sobre o Carro A, no mesmo critério, é de 1/3 e 1/7 respectivamente, ou seja, o inverso. Essa regra vale para a criação de todas as matrizes de julgamento. Na Tabela III vemos que "economia" é moderadamente (3) mais importante que "conforto" e fortemente (5) mais importante que "estilo". Já "conforto" é levemente (2) melhor que "estilo".

Após a criação das matrizes de julgamento, é necessário calcular as prioridades médias locais (PML's) e o vetor de prioridades dos critérios (VPC). Esse cálculo é feito somando os elementos de cada linha da matriz e dividindo pela soma de todos os elementos da matriz. Por fim, o cálculo das prioridades globais é efetuado, multiplicando os PML's pelo VPC. O vetor de prioridades global (PG), armazena a prioridade associada a cada alternativa em relação ao foco principal. A

Tabela IV ilustra os PML's, o VPC e o PG para o nosso problema.

Tabela IV. VETOR DE PRIORIDADES GLOBAL

	PML 1	PML 2	PML 3		VPC		PG
A	0.5889	0.1211	0.1615		0.6414		0.4260
B	0.3391	0.6414	0.0813	x	0.2375	=	0.3797
C	0.0720	0.2375	0.7572		0.1211		0.1943

De acordo com a Tabela IV, o carro A teve o maior indíce no vetor de prioridades global (PG). Ou seja, em uma escala de 0 a 1, ele satisfaz 0.4260 dos anseios da família. Desse modo ele será o carro escolhido para compra.

IV. MODELAGEM DO SISTEMA

Uma típica arquitetura de aplicação Web hospedada na nuvem é mostrada na Figura 2. Nela, podemos notar um *Front-End* que é responsável por receber as requisições dos clientes e distribuí-las entre os servidores no *Back-End*. Cada servidor é uma máquina virtual (*Virtual Machine - VM*) alugada junto a um provedor de infraestrutura na nuvem. Para modelar a arquitetura apresentada utilizamos um modelo de fila $M/M/m$, baseado na teoria de filas [23].

O problema a ser resolvido é encontrar o número mínimo de servidores (m) para atender as requisições que chegam ao sistema, dentro do tempo limite prometido aos clientes. A ferramenta utilizada para atingir este objetivo é chamada de autoscaling. De acordo com a Figura 2, no passo 1, os clientes enviam requisições para o sistema. As requisições são recebidas por um balanceador de carga, que as distribuem entre os servidores virtuais. Em seguida, no passo 2, as requisições são registradas em um arquivo ou banco de dados, chamado de *log*. Em 3, o módulo de previsão de demanda do autoscaling extrai as informações do *log*, cria uma série temporal e faz a previsão de demanda para o próximo intervalo de tempo. Na sequencia, em 4, a previsão é passada ao módulo de alocação de recursos, que calcula a quantidade de recursos necessária para atender a demanda e, no passo 5, solicita esses recursos ao provedor de infraestrutura na nuvem.

O autoscaling apresentado na Figura 2 é proativo, pois ele trabalha prevendo a demanda, a cada intervalo de tempo predefinido, para alocar os recursos necessários antes que o sistema fique sobrecarregado. O intevalo de tempo, também chamado de intervalo de reconfiguração, em que o autoscaling atua para alocar ou desalocar recursos foi configurado para uma hora. Esse valor foi escolhido porque a maioria dos provedores de infraestrutura na nuvem cobram pelos recursos alugados considerando o tempo mínimo de uma hora de uso. Assim sendo, para aproveitar ao máximo o investimento, é necessário planejar os recursos que serão usados a cada hora.

A. Previsão da demanda

Para prever a demanda é necessário utilizar algum modelo de previsão de séries temporais. O problema é que escolher esse modelo não é uma tarefa fácil, já que essa escolha está intimamente ligada as características da série em questão. Além disso, em uma aplicação Web, é comum existir variações nas características da demanda ao longo do tempo, o que faz com que um modelo que estava prevendo bem, passe a não ser mais adequado.

Para resolver o problema citado anteriormente propomos, neste trabalho, o uso de uma combinação linear de modelos, utilizando o método de análise hierárquica (AHP) para atribuir pesos adequados a cada modelo. Consideramos cinco modelos em nosso trabalho: NAIVE; AR(1); ARMA(1,1); ARIMA; e ETS. Para implementar esses modelos nós usamos as funções *arima(p, d, q)*, *auto.arima()* e *ets()* do pacote *forecast* [22] do software estatístico R.

Nosso objetivo é encontrar a importância de cada modelo de previsão (alternativa) de acordo com os critérios de subprovisionamento e superprovisionamento de recursos. A melhor solução será a qual minimizar o valor desses dois critérios. Recursos subprovisionados (provisionados a menos do que o necessário) fazem com que o servidor não consiga atender todas as requisições dentro do tempo estabelecido nos acordos de níveis de serviço, o que gera multas e má reputação para a empresa. Por outro lado, recursos superprovisionados (provisionados a mais do que o necessário) geram gastos desnecessários para a empresa. O objetivo é encontrar o vetor de prioridades global (PG) que soluciona o problema, e, então, determinar o peso de cada modelo de previsão.

A cada intervalo de tempo (uma hora), criamos as matrizes de julgamento e calculamos os PML's e o PG. Para definir a relação de prioridade entre os modelos em cada critério, usamos a seguinte regra: a cada intervalo de tempo é calculada a quantidade de recursos sub e superprovisionados gerados por cada modelo, de acordo com suas previsões passadas; em seguida, os modelos são ordenados em ordem crescente, de acordo com a quantidade de recursos (sub ou superprovisionados, de acordo com o critério que está sendo considerado) que geraram até o momento. O modelo que estiver na primeira posição tem prioridade 3 (moderada) sobre o segundo, 5 (alta) sobre o terceiro, 7 (muita alta) sobre o quarto, e 9 (extrema) sobre o quinto. O segundo tem prioridade 3 sobre o terceiro, 5 sobre o quarto e 7 sobre o quinto. O terceiro tem prioridade 3 sobre o quarto e 5 sobre o quinto. O quarto tem prioridade 3 sobre o quinto. Caso existam dois modelos empatados na mesma posição, terão prioridade 1 (igual) entre eles.

Para exemplificar, vamos imaginar que em determinado momento, os nossos cinco modelos de previsão (A, B, C, D e E) apresentem quantidades de recursos sub e superprovisionados de acordo com a Tabela V.

Tabela V. RECURSOS SUB E SUPERPROVISIONADOS

Recursos	A	B	C	D	E
Subprovisionados	15	18	12	20	25
Superprovisionados	10	14	10	21	16

De acordo com o cenário apresentado na Tabela V, as matrizes de julgamento ficaram como ilustradas na Tabela VI.

A partir das matrizes de julgamento mostradas na Tabela VI, calculamos os PML's e o PG, mostrado na Tabela VII. Consideramos, nesse exemplo, que os critérios subprovisionamento e superprovisionamento têm a mesma importância para o usuário, como mostra o VPC. Em seguida, usamos os valores do PG para combinar as previsões efetuadas por cada modelo. Ou seja, a previsão do modelo A será multiplicada por 0.3371, a do modelo B por 0.1568, e assim sucessivamente. Em seguida os valores resultantes são somados e, assim, teremos a previsão final para o próximo intervalo de tempo.

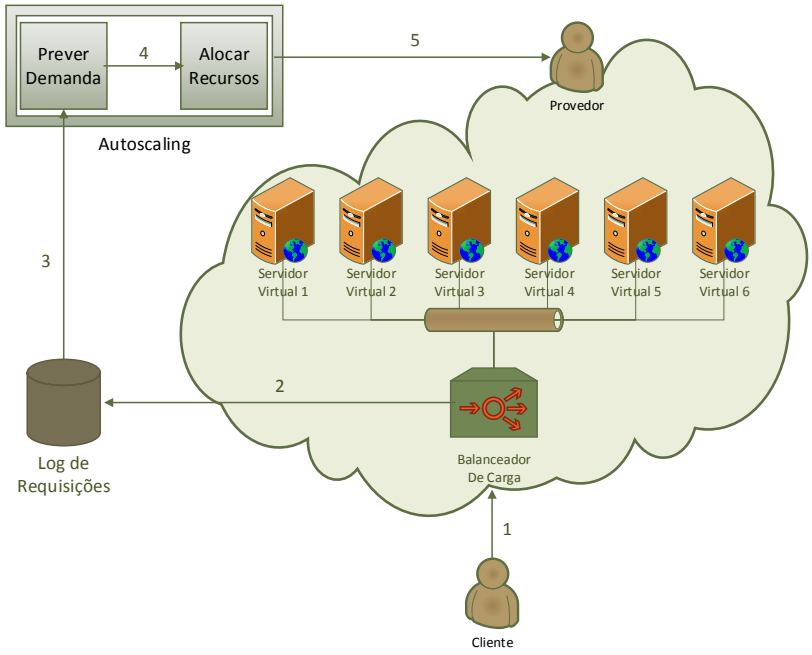

Figura 2. Aplicação Web hospedada na infraestrutura da nuvem.

Tabela VI. MATRIZES DE JULGAMENTO DOS MODELOS

Subprovisionados	A	B	C	D	E
A	1	3	1/3	5	7
B	1/3	1	1/5	3	5
C	3	5	1	7	9
D	1/5	1/3	1/7	1	3
E	1/7	1/5	1/9	1/3	1

Superprovisionados	A	B	C	D	E
A	1	5	1	9	7
B	1/5	1	1/5	5	3
C	1	5	1	9	7
D	1/9	1/5	1/9	1	1/3
E	1/7	1/3	1/7	3	1

Tabela VII. PESO DE CADA MODELO NA PREVISÃO (PG)

	PML Sub	PML Super		VPC		PG
A	0.2909	0.3833				0.3371
B	0.1636	0.1500	x	0.5	=	0.1568
C	0.4545	0.3833		0.5		0.4189
D	0.0728	0.0167				0.0447
E	0.0181	0.0667				0.0425

B. Alocação de recursos

Para determinar a quantidade de recursos a ser alocada, a partir da previsão de demanda, modelamos nosso sistema como uma fila $M/M/m$. Neste tipo de fila a utilização do sistema é modelada de acordo com a equação 1 [24].

$$\rho = \frac{\lambda}{m\mu} \tag{1}$$

Na equação 1, ρ é a utilização do sistema, λ é a taxa de chegada de requisições, μ é a taxa de processamento do servidor e m é o número de servidores.

Para o sistema ser estável, ρ precisa ser menor do que 1 [24]. Nosso objetivo é encontrar o menor valor para m que mantenha o sistema estável. Escrevendo a equação 1 em função de m temos a equação 2.

$$m = \lceil \frac{\lambda}{\rho\mu} \rceil \tag{2}$$

Para encontrar um valor adequado para ρ é necessário considerar o valor de μ e o tempo de resposta de uma requisição no servidor. O tempo de resposta do servidor, pode ser calculado como mostra a equação 3 [24]:

$$R = \frac{\frac{1}{\mu}}{1-\rho} \tag{3}$$

Colocando a equação 3 em função de ρ obtemos a equação 4.

$$\rho = 1 - \frac{1}{R\mu} \tag{4}$$

Juntando as equações 2 e 4 obtemos a equação 5, que representa o número de servidores, m, cada um capaz de processar μ requisições por segundo, necessário para atender uma taxa de chegada de λ requisições por segundo, com tempo de resposta máximo de R segundos [25].

$$m = \lceil \frac{R\lambda}{R\mu - 1} \rceil \quad (5)$$

C. Solução proposta

Para solucionar o problema apresentado, propomos, neste trabalho, combinar modelos de previsão de séries temporais, com o objetivo de otimizar a previsão da demanda e, consequentemente, a alocação de recursos. Consideramos cinco modelos de previsão: NAIVE; AR; ARMA; ARIMA; e ETS. Todos esses modelos foram descritos na seção III-A. Para atribuir pesos aos modelos de previsão, modelamos o nosso problema utilizando o AHP. No nosso caso temos dois objetivos a serem alcançados: minimizar o número de requisições não atendidas por falta de recursos (subprovisionamento); e minimizar o número de recursos ociosos (superprovisionamento). O algoritmo 1 ilustra o processo de alocação de recursos.

O algoritmo 1 recebe como entrada o log das requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R), a prioridade do critério subprovisionamento ($priSub$), e a prioridade do critério superprovisionamento ($priSuper$). A saída será a quantidade de recursos a ser alocada para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de recuros sub e superprovisionados. Na sequencia o algoritmo entra em um *loop* onde, a cada intervalo de tempo de uma hora, irá: atualizar o número de recursos sub e superprovisionados de acordo com as previsões passadas de cada modelo; criar as matrizes de julgamento, como descrito na seção III-B; calcular os PML's, como descrito na seção III-B; definir o peso de cada modelo na próxima previsão calculando o PG, como descrito na seção III-B; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão.

A atualização da quantidade de recursos sub e superprovisionados para cada modelo de previsão é feita da seguinta forma: supondo que o modelo A previu que, no próximo intervalo de tempo, seriam necessários 10 recursos (servidores virtuais) para atender a demanda, e o modelo B previu cinco. No entanto, ao final do intervalo, foi verificado que eram necessários oito recursos. Assim sendo, o modelo A irá acumular, em sua conta, dois recursos superprovisionados. Já o modelo B irá acumular três recursos subprovisionados. Essa conta é atualizada a cada hora.

V. Experimentos e Avaliação

Para realizar os experimentos, implementamos o algoritmo 1 usando a linguagem Java e simulamos a chegada de requisições com base em *logs* de servidores Web reais. A biblioteca *RJava* [26] foi utilizada para se comunicar com o pacote estatístico R [27], onde estão implementadas as funções *arima(p, d, q)*, *auto.arima()* e *ets()*, utilizadas em nossa proposta. O modelo AR foi implementado por meio da função *arima(1,0,0)*, O modelo ARMA por meio da função *arima(1,0,1)*, o modelo ARIMA pela função *auto.arima()* e o ETS pela função *ets()*. O modelo NAIVE foi implementado usando a linguagem Java. O computador utilizado para executar os experimentos foi um Quad Core intel I5 3.2 GHz, 16 GB Ram, 64 bits.

A. Logs Utilizados

Nós avaliamos nossa proposta utilizando três *logs* extraídos de servidores Web reais: o *log* Fifa World Cup 98 [28]; o *log* Nasa [19]; e o *log* ClarkNet [29]. O *log* Fifa World Cup é composto por todas as requisições recebidas pelo site da copa do mundo de 1998, entre 30 de abril de 1998 e 26 de julho de 1998. Durante esse período o site recebeu 1.352.804.107 requisições. O *log* Nasa é composto por dois meses de requisições recebidas pelo servidor da NASA Kennedy Space Center na Florida. O *log* ClarkNet é composto por duas semanas de requisições recebidas pelo servidor ClarkNet, que é um site do Metro da área metropolitana de Baltimore-Washington. Para criar nossas séries temporais, extraímos dos *logs* citados, a quantidade máxima de requisições por segundo a cada hora. A Figura 3 ilustra as séries utilizadas.

B. Resultados

As Figuras 4, 6 e 8 ilustram os resultados obtidos, considerando as três séries utilizadas neste trabalho e quatro cenários distintos descritos na Tabela VIII. Para cada cenário foram executados experimentos considerando: a mesma prioridade entre recursos sub e superprovisionados (AHP(1,1)); forte prioridade para recursos subprovisionados (AHP(5,1)); e forte prioridade para recursos superprovisionados (AHP(1,5)). O eixo X representa o número de recursos subprovisionados e o eixo Y representa o número de recursos superprovisionados, de acordo com cada modelo de previsão.

Tabela VIII. Cenários considerados nos experimentos

Cenário	Taxa de processamento (μ)	Tempo de resposta (R)
Primeiro	10 requisições/segundo	0.4 segundo
Segundo	10 requisições/segundo	0.8 segundo
Terceiro	20 requisições/segundo	0.4 segundo
Quarto	20 requisições/segundo	0.8 segundo

Para analisar a otimalidade de uma solução multiobjetivo, como é o nosso caso, é utilizado o conceito de solução não dominada, ou ótima de Pareto, que é uma solução para a qual não existe outra solução viável que melhore, ao mesmo tempo, todos os objetivos. Todas as soluções com essas características pertencem ao conjunto ótimo de Pareto.

A Figura 4 mostra os resultados para a série Fifa World Cup. As soluções que estão sob a linha são as chamadas soluções ótimas de Pareto ou soluções não dominadas. A linha representa a fronteira ótima de Pareto. Note que a nossa proposta (AHP) está entre as soluções ótimas de Pareto em todos os cenários considerados, ao lado dos modelos NAIVE e AR. Percebemos que o AHP(5,1), como era de se esperar, teve menos recursos subprovisionados em relação ao AHP(1,1). Em compensação teve mais recursos superprovisionados. No caso do AHP(1,5) a situação foi a inversa: mais recursos subprovisionados e menos superprovisionados.

Os valores apresentados no gráfico ilustram o número de recursos sub e superprovisionados ao longo dos três meses de duração do *log*. Percebe-se que, o número de recursos, tanto sub como superprovisionados, caem gradativamente do

Algoritmo 1: ALOCAÇÃO DE RECURSOS

Entrada: Log das requisições, $\mu, R, priSub, priSuper$
Saída: Número de servidores virtuais (recursos) a serem alocados (m)

inicio
  $sub \leftarrow$ atribuir 0 para o valor de recursos subprovisionados de cada modelo
  $super \leftarrow$ atribuir 0 para o valor de recursos superprovisionados de cada modelo
  para *cada hora* **faça**
    $sub \leftarrow$ atualizar o número de recursos subprovisionados para cada modelo
    $super \leftarrow$ atualizar o número de recursos superprovisionados para cada modelo
    $mjSub \leftarrow$ criar a matriz de julgamento dentro do critério subprovisionamento usando sub
    $mjSuper \leftarrow$ criar a matriz de julgamento dentro do critério superprovisionamento usando $super$
    $mjCriterios \leftarrow$ criar a matriz de julgamento entre os critérios usando $priSub$ e $priSuper$
    $pmlSub \leftarrow$ calcular o PML para o critério subprovisionamento usando $mjSub$
    $pmlSuper \leftarrow$ calcular o PML para o critério superprovisionamento usando $mjSuper$
    $pmlCriterios \leftarrow$ calcular o PML entre os critérios usando $mjCriterios$
    $pg \leftarrow$ calcular o vetor de prioridades global usando $pmlSub$, $pmlSuper$ e $pmlCriterios$
    $ts \leftarrow$ criar série temporal a partir dos dados extraídos do log
    para *cada modelo de previsão i* **faça**
      $funcao_i \leftarrow$ estimar uma função matemática usando ts e o modelo i
      $prev_i \leftarrow$ prever a demanda para o próximo periodo usando a $funcao_i$
    fim
    $prev_f \leftarrow \sum_{i=1}^{5} prev_i pg_i$
    $m \leftarrow$ calcular número de recursos usando $\mu, R, prev_f$ por meio da equação 5
    alocar m recursos junto ao provedor de infraestrutura na nuvem
  fim
fim

primeiro para o quarto cenário. A explicação para isso está nos parâmetros (μ) e (R). Quanto maiores forem os valores desses parâmetros, menor será a quantidade necessária de recursos para atender a demanda.

A Figura 5 ilustra como foi a evolução dos pesos de cada modelo para o AHP(1,1), considerando o primeiro cenário, na série FIFA World Cup. podemos notar que os modelos NAIVE e AR(1) foram os modelos dominantes na composição da solução, principalmente a partir do tempo 500. Esses dois modelos apresentaram, respectivamente, o melhor resultado para recursos sub e superprovisionados. Isso explica porque ambos tiveram os maiores pesos na nossa solução.

Observando a Figura 6, que ilustra os resultados para a série Nasa, podemos notar que nossa solução (AHP) está novamente entre as melhores em todos os cenários. No entanto, no primeiro cenário, a solução AHP(1,1) foi dominada pela AHP(1,5), já que ambas tiveram o mesmo número de recursos subprovisionados mas a AHP(1,5) teve menos recursos superprovisionados. O modelo ARMA também foi dominado no primeiro cenário. No segundo cenário, os modelos AR, ARMA e ETS foram dominados. Já no terceiro e quarto cenários, todas as soluções, exceto o NAIVE, ficaram entre as soluções ótimas. Percebemos que, para a série Nasa, o número de recursos, tanto sub como superprovisionados, foi bem menor do que para a série Fifa World Cup. Isso se explica pelos seguintes fatos: o tempo de duração do *log* Nasa é de dois meses, enquanto o *log* FIFA World Cup é de três meses; e, ainda, a quantidade de requisições recebidas pelo site da Nasa foi bem inferior a do site Fifa World Cup, durante o período considerado. Isso pode ser observado na Figura 3.

Observando a Figura 7, que mostra a evolução dos pesos de cada modelo para o AHP(1,1), considerando o primeiro cenário, na série Nasa, podemos notar que os modelos NAIVE, AR(1) e ARMA(1,1) dominaram a composição da solução na primeira metade da série. Na segunda metade o modelo ARMA(1,1) perdeu força e o modelo ARIMA passou a ter uma participação maior.

Para concluir, a Figura 8 mostra os resultados para a série ClarkNet. Assim como nos casos anteriores, a nossa proposta (AHP) está no conjunto de soluções ótimas de Pareto, em todos os cenários testados. No entanto, no primeiro e no segundo cenários, a solução AHP(5,1) foi dominada pela solução AHP(1,1). Já no terceiro cenário, as soluções AHP(5,1) e AHP(1,1) foram dominadas pela solução AHP(1,5). Isso mostra que, o fato de atribuir maior prioridade para determinado critério não implica, necessariamente, que esse critério será melhorado na solução. De acordo com os resultados, notamos que existe um limite até onde determinado critério pode ser melhorado em determinado cenário. Por exemplo, no terceiro cenário, não adiantou aumentar a prioridade do critério subprovisionamento de recursos por meio da solução AHP(5,1), pois isso não foi sufuciente para melhorar esse critério em relação a solução AHP(1,5). Os resultados mostram, para está série, o menor número de recursos sub e superprovisionados, comparando com as séries anteriores. A explicação para isto é que o log ClarkNet tem a mais curta duração entre todos: apenas duas semanas.

Na Figura 9 podemos observar a evolução dos pesos de cada modelo para o AHP(1,1), considerando o primeiro cenário, na série ClarkNet. O modelo AR teve grande importância no início da série, mas a partir do tempo 100 perdeu importância. A partir desse ponto o modelo ETS passou a

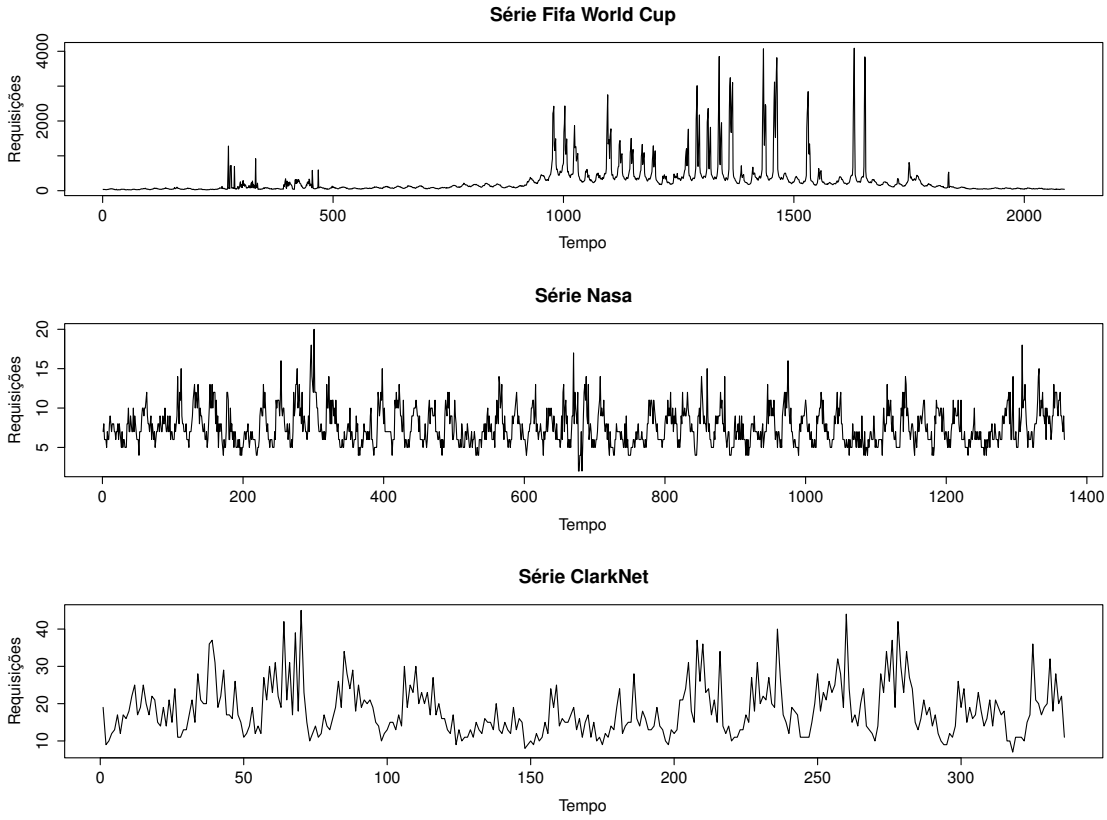

Figura 3. Séries Utilizadas.

ganhar mais peso. O modelo ARMA teve altos e baixos ao longo da série. Já o modelo NAIVE teve relativa importância durante quase toda a série, mas no final teve uma queda considerável. Notamos que houve uma grande variação nos pesos dos modelos ao longo da série. Isso mostra a capacidade da nossa solução em se adaptar a mudanças na série.

O *overhead* gerado pela nossa proposta para fazer uma previsão, considerando o ambiente descrito no início desta seção, foi de, aproximadamente, 400 milisegundos para uma série com 300 pontos de dados. Esse *overhead* é desprezível, considerando que será feita apenas uma previsão a cada hora. Comparando com o *overhead* gerado por cada modelo individualmente, o modelo ETS demora pouco mais de 250 milisegundos para fazer uma previsão, considerando uma série com 300 pontos de dados. O modelo ARIMA leva em torno de 100 milisegundos. O modelo ARMA(1, 1) gasta em torno de 8 milisegundos e o modelo AR(1) em torno de 5 milisegundos. No caso do modelo NAIVE, praticamente não existe *overhead*.

VI. Conclusão e trabalhos futuros

O objetivo deste trabalho foi realizar um estudo sobre a utilização do método de análise hierárquica (AHP) para combinar modelos de previsão de séries temporais, visando otimizar a alocação de recursos para aplicações Web hospedadas na nuvem. A previsão de recursos é fundamental para evitar instabilidades no sistema, uma vez que técnicas reativas, devido ao *delay* existente entre a solicitação e a entrega de um recurso pelo provedor de nuvem, não são capazes de lidar com a variação de demanda a tempo de evitar que o sistema fique sobrecarregado. Os resultados mostraram que não existe um modelo de previsão que seja o melhor em todos os casos. Assim sendo, nossa proposta se mostrou mais eficiente do que usar um único modelo de previsão, sendo capaz de se adaptar a diversas aplicações, com diferentes tipos de demanda. Ainda, de acordo com os resultados apresentados, nossa proposta (AHP) foi a única que, considerando todos os cenários e séries avaliadas, nunca foi dominada por nenhum modelo isolado, estando sempre entre as soluções ótimas de Pareto. Isso mostra sua capacidade de adaptação a diferentes tipos de séries e diferentes cenários, convergindo sempre para uma solução ótima. Como trabalhos futuros, pretendemos considerar mais modelos de previsão, mais tipos de séries temporais e mais cenários para a avaliação. Também, pretendemos comparar essa proposta com outras técnicas de otimização multiobjetivo.

Agradecimentos

Gostariamos de agradecer o Instituto Federal de São Paulo (IFSP) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro. Os logs usados neste trabalho foram obtidos no site http://ita.ee.lbl.gov/html/traces.html.

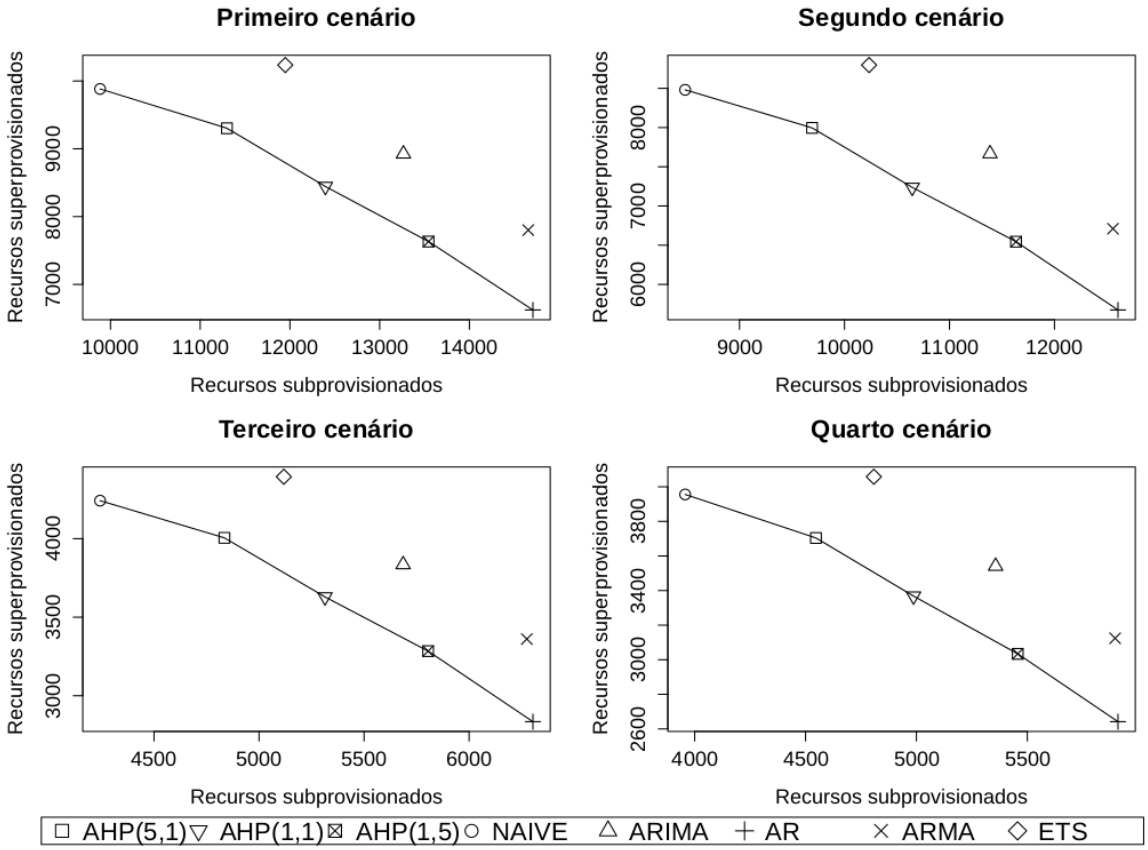

Figura 4. Soluções para a série Fifa World Cup.

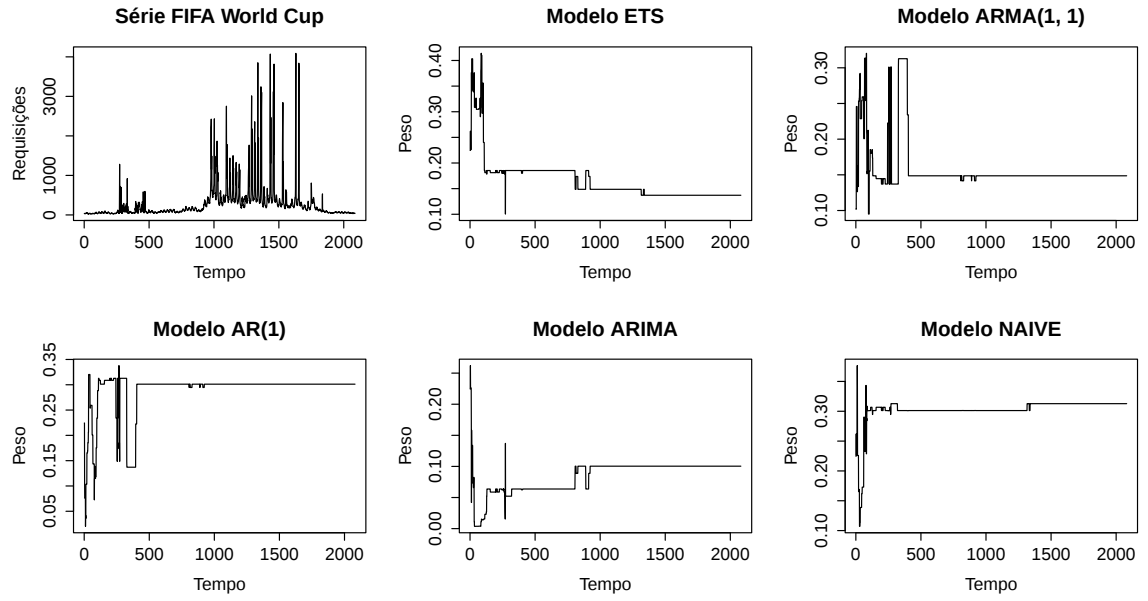

Figura 5. Peso de cada modelo ao longo do tempo para o AHP(1,1) na série FIFA World Cup

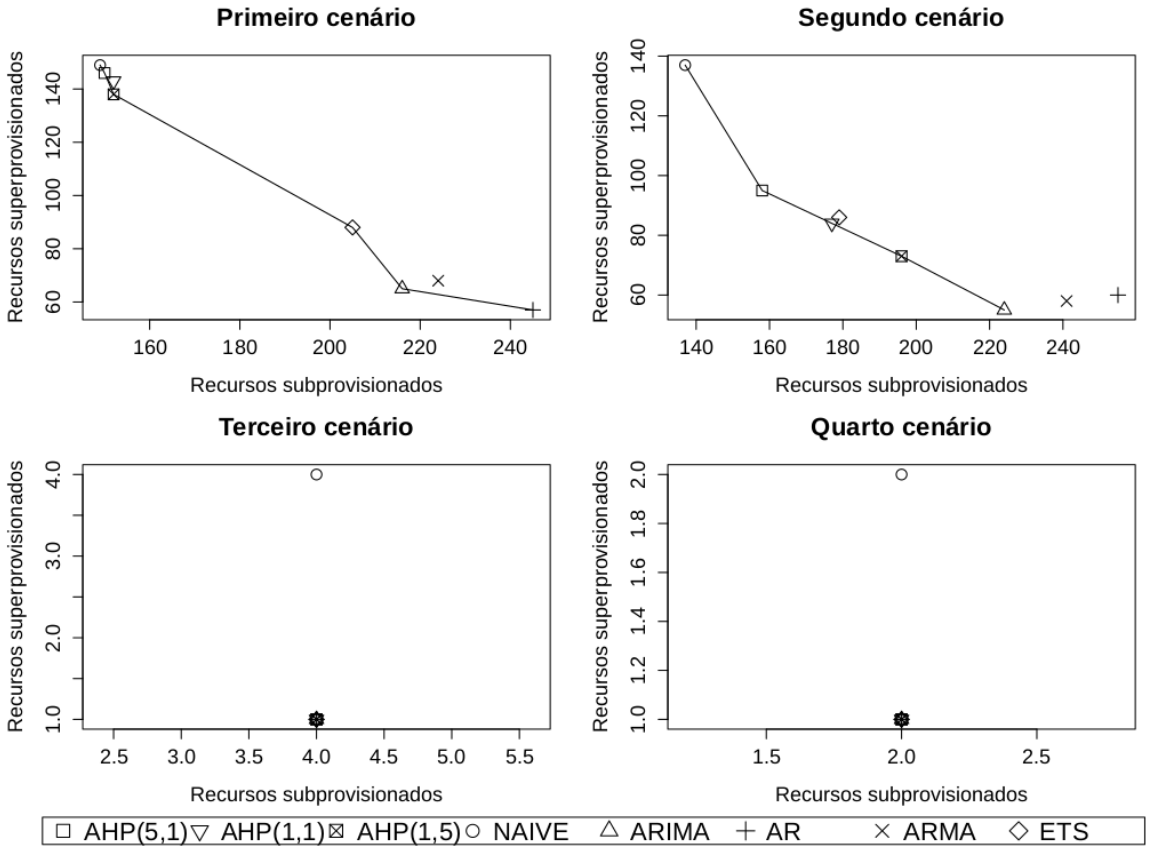

Figura 6. Soluções para a série Nasa.

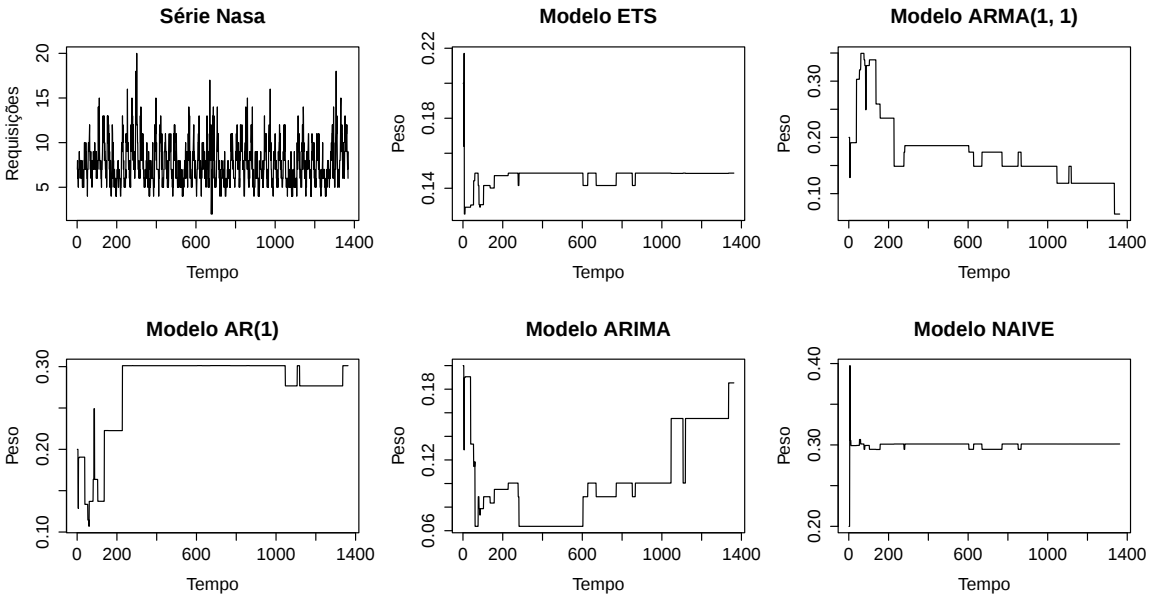

Figura 7. Peso de cada modelo ao longo do tempo para o AHP(1,1) na série Nasa

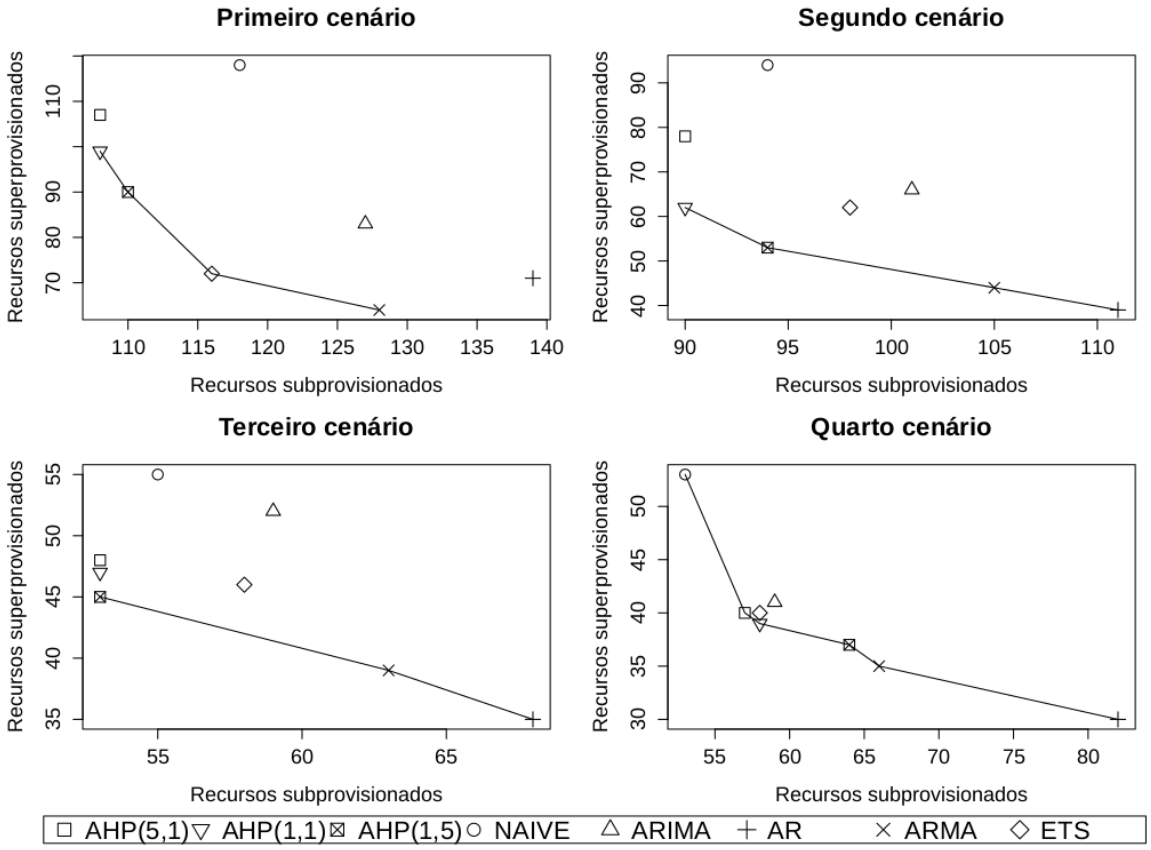

Figura 8. Soluções para a série ClarkNet.

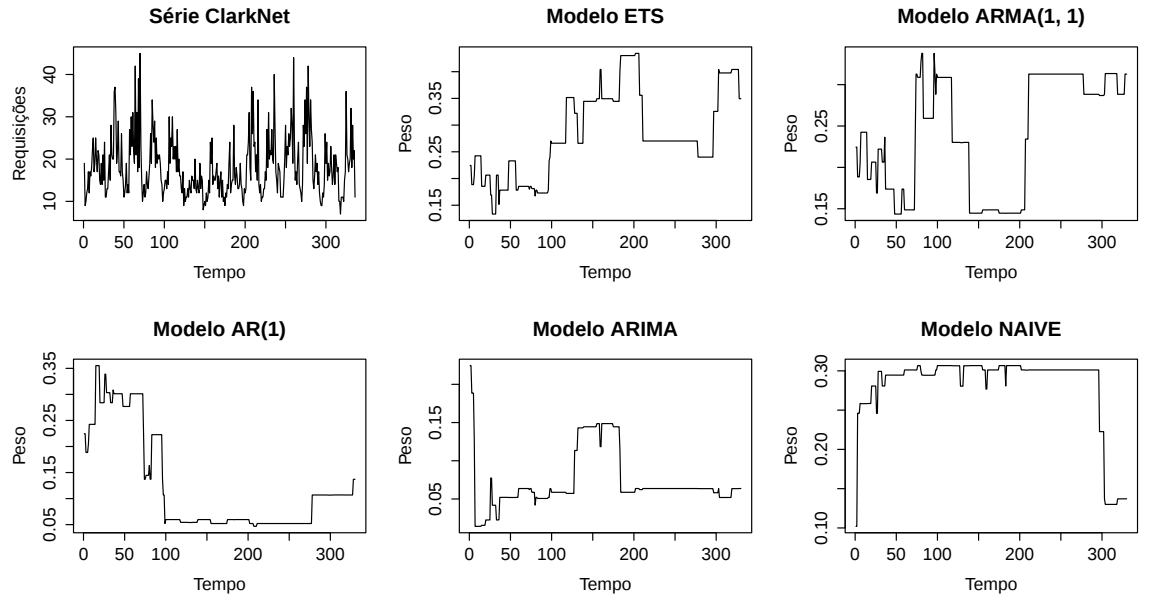

Figura 9. Peso de cada modelo ao longo do tempo para o AHP(1,1) na série ClarkNet

REFERÊNCIAS

[1] M. Armbrust, O. Fox, R. Griffith, A. D. Joseph, Y. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "M.: Above the clouds: A berkeley view of cloud computing," 2009.

[2] M. Miller, *Cloud computing: Web-based applications that change the way you work and collaborate online*. Que publishing, 2008.

[3] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "Auto-scaling techniques for elastic applications in cloud environments," *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09*, vol. 12, 2012.

[4] T. L. Saaty, *What is the analytic hierarchy process?* Springer, 1988.

[5] T. Clark, "Quantifying the benefits of the rightscale cloud management platform," *Fact Point Group Whitepaper, funded by Rightscale*, 2010.

[6] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM, 2010, p. 4.

[7] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities," in *Proceedings of the 1st workshop on Automated control for datacenters and clouds*. ACM, 2009, pp. 13–18.

[8] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 204–212.

[9] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proceedings of the 4th ACM European conference on Computer systems*. ACM, 2009, pp. 13–26.

[10] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters," in *Proceedings of the 6th international conference on Autonomic computing*. ACM, 2009, pp. 117–126.

[11] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. A. Fortes, "Fuzzy modeling based resource management for virtualized database systems," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*. IEEE, 2011, pp. 32–42.

[12] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "On the use of fuzzy modeling in virtualized data center management," in *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*. IEEE, 2007, pp. 25–25.

[13] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 58–65.

[14] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 500–507.

[15] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *Network, IEEE*, vol. 14, no. 3, pp. 30–37, 2000.

[16] H. Fernandez, G. Pierre, T. Kielmann *et al.*, "Autoscaling web applications in heterogeneous cloud infrastructures," in *IEEE International Conference on Cloud Engineering*, 2014.

[17] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency and Computation: Practice and Experience*, 2014.

[18] M. Balaji, G. Rao, C. Kumar *et al.*, "A comparitive study of predictive models for cloud infrastructure management," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 923–926.

[19] "Nasa-http - two months of http logs from the ksc-nasa www server," http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, accessed: 2014-10-15.

[20] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[21] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[22] R. J. Hyndman and Y. Khandakar, "Automatic time series for forecasting: the forecast package for r," Monash University, Department of Econometrics and Business Statistics, Tech. Rep., 2007.

[23] M. Di Penta, G. Casazza, G. Antoniol, and E. Merlo, "Modeling web maintenance centers through queue models," in *Software Maintenance and Reengineering, 2001. Fifth European Conference on*. IEEE, 2001, pp. 131–138.

[24] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2013.

[25] D. A. Menascé, V. A. Almeida, and L. W. Dowdy, *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall PTR Upper Saddle River, 2002.

[26] S. Urbanek and M. S. Urbanek, "Package rjava," 2014.

[27] J. Braun and D. J. Murdoch, *A first course in statistical programming with R*. Cambridge University Press Cambridge, 2007, vol. 25.

[28] "1998 world cup web site access logs," http://ita.ee.lbl.gov/html/contrib/WorldCup.html, accessed: 2014-10-15.

[29] "Clarknet-http - two weeks of http logs from the clarknet www server," http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html, accessed: 2014-10-15.