Controle do Consumo de Energia em *Data Centers* via Aprendizado por Reforço

João Gabriel Silva Marra

JUIZ DE FORA

MARÇO, 2016

Controle do Consumo de Energia em *Data Centers* via Aprendizado por Reforço

João Gabriel Silva Marra

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Saul Castro Leite

JUIZ DE FORA

MARÇO, 2016

Controle do Consumo de Energia em *Data Centers* via Aprendizado por Reforço

João Gabriel Silva Marra

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Saul Castro Leite
Professor Doutor

Leonardo Goliatt
Professor Doutor

Heder Bernardino
Professor Doutor

JUIZ DE FORA

4 DE MARÇO, 2016

Agradeço a Deus, pela força e coragem durante toda esta longa caminhada.

Aos meus pais, Moema e Benedito, pelo sustento, apoio e por acreditarem em mim durante todos os longos anos de graduação.

Aos meu irmãos Lucas, Paulo e Pedro pela amizade e companheirismo.

A minha namorada Isabella Flores, pelo carinho e apoio, e por estar sempre perto nos momentos difíceis ajudando a resolver todos os problemas.

Aos meus colegas de GET, e tutores Flávia Bastos e Heder Bernardino por terem me mostrado o caminho certo a seguir na área acadêmica. Ao meu orientador Saul, por ter me ajudado a descobrir a ciência e me apronfudar nela. Aos demais professores dos departamentos de Ciência da Computação e Mecânica Aplicada e Computacional, cada um me proporcionando o conhecimento necessário para ultrapassar cada etapa.

Aos meus colegas, amigos e companheiros de curso, Tales Lima, João Marcos, Jonata Jefferson, Lucas Berg, Viviane Galvão, por estarem presentes em cada trabalho, exercício e prova.

Resumo

Atualmente a alta demanda por serviços de Internet cria a necessidade de operar sistemas de processamento paralelo com um grande número de máquinas. Muitas vezes tais sistemas operam com um número de servidores maior do que seria necessário para garantir serviço de forma adequada em momentos de pico de demanda. Contudo, isso gera um enorme gasto de energia que poderia ser evitado. Neste trabalho é proposto um método de controle do sistema de servidores utilizando aprendizado por reforço, visando achar uma maneira para minimizar os gastos. O sistema considerado aqui opera com um bloco de servidores que estão sempre em funcionamento e um ou mais blocos de servidores reservas que podem ser ligados ou desligados. O algoritmo Q-learning é usado para determinar o melhor momento para ligar os blocos de servidores reservas. Uma simulação foi desenvolvida para testar este método.

Palavras-chave: teoria de filas, aprendizado por reforço, data center

Abstract

Nowadays the high demand for Internet services creates the need for systems with parallel processing and a large number of machines. Often, these systems operate with more servers than needed to ensure an appropriate service at moments of peak demand. However, this generates enormous energy consumption which could be avoided. This work proposes a method to control the energy consumption using reinforcement learning, aiming to find a better way to minimize the total cost while maintaining quality of service. The system considered works with one block of servers, called regular, which is always turned on and one or more blocks, called reserves, which can be turned on or off. The Q-learning algorithm is used to determine the best moment to turn on or off the reserve blocks. A simulator was developed to test the proposed method.

Keywords: queueing theory, reinforcment learning, data center

Sumário

Lista de Figuras **5**

Lista de Tabelas **6**

Lista de Abreviações **7**

1 Introdução **8**

2 Trabalhos Relacionados e Métodos Empregados **11**
2.1 Métodos Empregados 13
2.1.1 Processo de Decisão de Markov 13
2.1.2 Q-learning 15
2.1.3 Processo de Decisão semi-Markoviano(SMDP) 17

3 Simulador **20**
3.1 Simulação a Eventos Discretos 20
3.2 Modelo do sistema 20
3.3 Implementação 22

4 Resultados das Simulações **25**
4.1 Base de Testes 25
4.2 Análise dos Resultados 26

5 Conclusões **32**

Referências Bibliográficas **33**

Lista de Figuras

2.1 Esquema de funcionamento de uma Cadeia de Markov 14

3.1 Representação do *server farm* 21
3.2 Representação da matriz Q 21
3.3 Ilustração do funcionamento de uma fila de eventos hipotética e do sistema de controle da simulação 24

4.1 Evolução do número de servidores ligados para o sistema de controle com um bloco reserva 29
4.2 Evolução do número de servidores ligados para o sistema de controle com quatro blocos reservas 30
4.3 Evolução do número de servidores ligados para o sistema de controle com oito blocos reservas 31

Lista de Tabelas

4.1 Número de servidores e blocos reservas em cada base de teste 26
4.2 Resultados dos testes - média de servidores ligados 27
4.3 Resultados dos testes - variância de servidores ligados 28
4.4 Resultados dos testes - média de clientes no sistema 28
4.5 Resultados dos testes - variância de clientes no sistema 28
4.6 Resultados dos testes - média do custo do sistema 28

Lista de Abreviaçõs

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
DVFS	Dinamic Voltage and Frequency Scaling
MDP	Markov Decision Processes
SMDP	semi-Markov Decision Processes
FIFS	First In First Served

1 Introdução

Não é de hoje que o mundo depende de serviços pela Internet. Estima-se que o número usuários da Internet ultrapasse 3,2 bilhões (União Internacional das Telecomunicações - UIT, 2014). A grande maioria destes usuários utilizam os serviços que estão disponíveis na Internet, como as ferramentas de buscas (como o *Google*, *Yahoo*, *Bing*, dentre outras), as redes sociais (como o *Facebook* e outros), e os canais de streaming de videos (como o *Youtube*, *Netflix*, dentre outros). Além disso, com a disponibilidade cada vez maior de acesso de Banda Larga à Internet, existe a tendência cada vez maior do surgimento e utilização de aplicações que executam remotamente. Alguns exemplos são gerenciadores de email, como o *Gmail*, *Yahoo mail* e *Hotmail*, processadores de texto, tais como o *Google docs*, e até aplicações mais avançadas como o *WolframAlpha*, que consegue fazer computação simbólica. Estes são exemplos do que vem sendo chamado de Computação nas Nuvens (*Cloud Computing*) que é definida pelo uso da capacidade de armazenamento e processamento de computadores e servidores compartilhados através da Internet. Todos estes serviços dependem das chamadas fazendas de servidores (*server farm*) que recebem e processam toda esta demanda. *Server farms*, são aglomerados de máquinas conectadas em rede e funcionam como sendo um único servidor. Geralmente as tarefas enviadas ao *server farm* são divididas e processadas em paralelo pelos computadores que formam o aglomerado. Isto é feito para atender toda a demanda e dar uma resposta rápida ao usuário. Para suportar toda esta demanda, serviços cada vez mais complexos e poder atuar no mundo inteiro são necessárias milhares de máquinas por sistema.

Além disso essas máquinas possuem um alto desempenho e geralmente ficam ligadas constantemente para atender eventuais picos de demanda. Um agravante é que servidores ociosos consomem cerca de 65% do total de energia se estiverem em processamento, mesmo servidores com componentes mais atuais (GREENBERG, 2008). Nos Estados Unidos, em 2013, o consumo de energia dos *data centers* foi de cerca de noventa e um bilhões de kilowatts-hora, energia suficiente para abastecer todas as casas de Nova York duas vezes pelo mesmo período (NATURAL RESOURCES DEFENCE COUNCIL

- NRDC, 2014) e estima-se que mais de 1,5% do total do consumo de energia nos Estados Unidos vem de *data centers* (GANDHI et al., 2010). Esse consumo vem não somente do processamento das máquinas mas também dos grandes sistemas de refrigeramento. Esse gasto para manter a temperatura das máquinas fez com que, em 2014, o *Facebook* decidisse criar um de seus *data centers* no Ártico para que o clima local ajudasse a controlar todo o calor gerado pelo alto processamento. Todo este consumo de energia gera grandes gastos financeiros para as empresas que disponibilizam estes serviços e também gera grande impácto ambiental. Neste sentido, é preciso traçar estratégias para que o consumo de energia seja diminuído sem afetar o desempenho do sistema.

Uma forma de reduzir o consumo de energia destes sistemas é desligando algumas máquinas do aglomerado em momentos de baixa demanda e religando as máquinas em momentos de alta demanda. Uma dificuldade nesta estratégia é encontrar os momentos certos para desligar e ligar as máquinas pois existe um tempo, geralmente chamados de *setup time*, em que os computadores consomem energia sem conseguir processar requisições.

Neste trabalho, é proposto a utilização de métodos de aprendizado por reforço para determinar o melhor momento de mudar as máquinas de estado. Métodos de aprendizado por reforço são capazes de determinar uma aproximação da política ótima dentro de um cenário de minimização do custo total esperado com desconto. Estes métodos possuem a vantagem de não depender de hipóteses sobre a entrada das tarefas no sistema e não dependem de um modelo matemático para o sistema de filas. A proposta é dividir os servidores em blocos, onde existirá um bloco, chamado de regular, que ficará sempre ligado, e um ou mais blocos, chamados de reservas, que poderão ser ligados ou desligados conforme a demanda. Para testar o método proposto, foi implementado uma simulação de um *server farm* que utiliza métodos de aprendizado por reforço para determinar uma política para ligar e desligar os servidores.

O restante do trabalho está dividido nos seguintes capítulos: no Capítulo 2, será discutido outras abordagens encontradas na literatura para tratar o problema mencionado acima e será feito uma revisão dos métodos que serão empregados; no Capítulo 3, será mostrado como a simulação foi implementada; nos Capítulos 4 e 5, serão mostrados os

resultados da simulação e as conclusões.

2 Trabalhos Relacionados e Métodos Empregados

O problema de controlar o consumo de energia em *server farms* é abordado de diferentes formas na literatura. Este capítulo apresenta uma revisão sobre estas abordagens e compara com a proposta deste trabalho.

Existem autores que propõem uma abordagem baseada no controle da voltagem (ou frequência) em que os processadores trabalham (*Dynamic Voltage or Frequency Scaling* - DVFS). Em uma frequência menor de processamento o servidor consome menos energia e produz menos calor. Um exemplo desta abordagem é vista em (LEE e KULKARNI, 2014), em que esta proposta é utilizada para controlar o custo de *data centers* onde a taxa de entrada de tarefas no servidor é controlada. Contudo, E. LE SUEUR e G. HEISER (2010) fazem uma análise da evolução na tecnologia de economia de energia dos processadores atuais e concluem que os processadores e memórias atuais proporcionam um menor alcance no controle DVFS. Além disso, os autores argumentam que em alguns casos o uso de DVFS pode aumentar o consumo de energia.

Outros autores propõem um controle onde é possível desligar ou colocar os servidores em um estado de hibernação. Esta abordagem é considerada em (GANDHI et al., 2010), (M. MAZZUCCO e D. DYACHUK, 2012), (NYIATO et al., 2009), (ARTALEJO et al., 2005), (MITRANI, 2013). GANDHI et al. (2010) propõem, além de ligar e desligar servidores, o uso de vários estágios de hibernação, dividindo a quantidade de servidores em várias partes que podem ter estágios de hibernação diferentes. Os autores concluem que políticas mistas em que vários estágios de hibernação são possíveis não acarretam em ganhos. Essa proposta leva em conta que o tempo de configuração (tempo onde o servidor esta ligando/desligando) pode afetar muito o desempenho do sistema. ARTALEJO et al. (2005) propõem um modelo de fila que só consegue ligar ou desligar um servidor por vez. Seus objetivos são estudar a distribuição estacionária do estado do sistema utilizando uma equação diferencial, investigar a distribuição do tempo de espera de um cliente e analisar

os períodos onde o sistema está mais ocupado e os maiores tamanhos de filas alcançados. NYIATO et al. (2009) utilizam um modelo onde não existe fila de clientes. As requisições chegam a um sistema denominado *Job Broker* que é usado para distribuir estas requisições para diversos *server farms*. Cada *server farm* utiliza um sistema de *batch scheduling* que divide as requisições nos servidores. NYIATO et al. (2009) modelam o problema como um Processo de Decisão de Markov (MDP do inglês *Markov Decision Processes*) a tempo discreto. Em (M. MAZZUCCO e D. DYACHUK, 2012), os autores introduzem uma análise aproximada de um *data center* com muitos servidores, que podem ser ligados e desligados, e usam uma estimativa de tráfego pesado, com uma probabilidade de perder clientes caso os servidores estejam ocupados. A maioria dos artigos citados assumem como hipótese para seus modelos que o tempo entre entradas e serviços no sistema possuem distribuição exponencial. Isso nem sempre corresponde à realidade, já que existe uma alta variação no processo de entrada para este tipo de serviço, o que não é refletido com esta hipótese.

Em (MITRANI, 2013) é estudado um tipo de controle baseado em blocos. Ele trabalha com apenas dois blocos: um bloco regular, que está sempre ligado, e um bloco reserva, que pode ser ligado e desligado. Seu controle é feito a partir de dois limiares pré estabelecidos. Estes limiares correspondem à quantidade de usuários na fila do sistema. Se esse número ultrapassa o primeiro limiar, que é definido como limiar superior, o bloco reserva é ligado para tratar o excesso de requisições do sistema. Quando o sistema volta a um estado onde o número de clientes na fila fica menor que o segundo limiar, chamado limiar inferior, o bloco reserva é desligado. Utilizando um modelo matemático para o sistema de fila com uma fila única para múltiplos servidores, MITRANI (2013) chegou a um valor para os limiares superior e inferior relativo ao número de servidores no bloco reserva. Para obter o seu modelo matemático de fila, MITRANI (2013) assumiu que o tempo para entradas consecutivas é dado por uma distribuição exponencial.

Neste trabalho é abordado um sistema de controle semelhante à proposta por (MITRANI, 2013). O controle é feito também por blocos de servidores, mas com um bloco de servidores regulares (sempre ligados) e um ou mais blocos de servidores reservas. Diferentemente do que foi feito em (MITRANI, 2013), neste trabalho não é utilizado um modelo de filas para determinar a política de controle, já que os métodos de aprendizado

por reforço não dependem disto. Desta forma, não é preciso assumir a distribuição do tempo entre a entrada de clientes consecutivos. Assim, o método estaria apto a tratar entradas com alta variação, que não são capturadas com a hipótese de distribuição exponencial. Além disso, dividir os servidores reservas em mais de um bloco proporciona maior versatilidade ao controle, já que é possível atender a níveis intermediários de demanda (não somente uma baixa demanda e a uma alta demanda). Portanto serão ligados apenas a quantidade de blocos necessária para atender aquela demanda.

Em MARRA et al. (2013) foi utilizado o mesmo método empregado neste trabalho para fazer o controle em um sistema onde existe apenas um bloco reserva.

2.1 Métodos Empregados

Neste trabalho foram utilizados métodos de aprendizado por reforço para fazer o controle nos blocos de servidores reservas. Estes métodos são capazes de encontrar uma aproximação de uma política ótima para o controle dos servidores sem a necessidade de um modelo matemático para a fila e uma hipótese para o tempo entre entradas de clientes e processamento de tarefas. Os métodos de aprendizado por reforço são derivados dos Processos de Decisão de Markov.

2.1.1 Processo de Decisão de Markov

Um processo de decisão de Markov é um processo estocástico a tempo discreto $\{X_n\}$ que toma valor em um certo conjunto E, em que, para cada $n \in \mathbb{N}$, X_n representa o estado de um certo sistema de interesse. A cada instante n, é possível interferir nesse sistema através de uma ação a, pertencente a um certo conjunto de possíveis ações A. Considere ainda que, se o sistema se encontra no estado $i \in E$ no instante n, ele transita para um certo estado $j \in E$ no tempo $n+1$ somente dependendo da ação escolhida e os estados i e j envolvidos na transição. Ou seja, é possível descrever a evolução deste sistema através de certas probabilidades de transição $p_{ij}(a)$ dependendo somente de a, i e j, onde $i, j \in E$ e $a \in A$. A Figura 2.1 ilustra o grafo de transição de uma cadeira de Markov. Nela é possível ver que a transição de um estado i para outro estado j depende da probabilidade

p_{ij}, por exemplo, se o sistema se encontra no estado $E1$ ele terá a probabilidade p_{13} de ir para o estado $E3$. Nota-se que no problema descrito acima, as probabilidades de transição p_{ij} podem se alterar dependendo da ação $a \in A$ escolhida. Desta forma, dependendo da ação escolhida, o grafo de transição pode ser alterado.

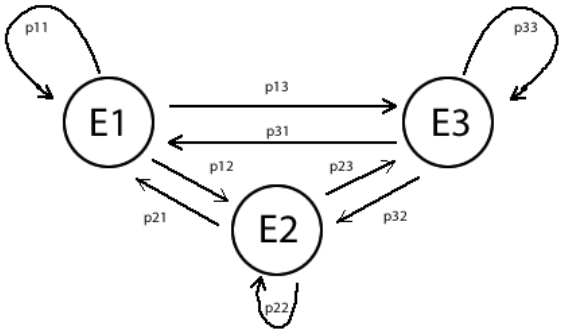

Figura 2.1: Esquema de funcionamento de uma Cadeia de Markov

Geralmente, é desejado determinar uma sequência de ações $a_1, a_2, a_3, \ldots$, geradas por uma política que minimiza uma certa medida de custo do sistema. Seja $r(i, a)$ o custo imediato obtido quando o sistema está no estado i e a ação a é aplicada. Suponha que estas políticas são determinadas por regras de decisão $\pi : E \to A$, onde para cada estado i, $\pi(i)$ determina a ação que deve ser tomada quando o processo se encontra em tal estado. É desejado escolher uma política π que minimiza o seguinte custo:

$$w^{\pi}(i) = \mathbb{E}_i^{\pi}\left[\sum_{n=0}^{\infty} \lambda^n r(X_n, \pi(X_n))\right],$$

onde $i \in E$ representa o estado inicial (i.e., $X_0 = i$) e a constante $\lambda \in (0, 1)$ é um fator de desconto. É dito "de Markov" pois obedece à *propriedade de Markov*: o efeito de uma ação em um estado depende apenas da ação e do estado atual do sistema independente de como ele chegou a este estado.

Um resultado importante sobre a teoria de MDP é que o valor do custo mínimo $v^*(i) = \inf_{\pi} w^{\pi}(i)$ pode ser determinado através do algoritmo da Iteração de Valor (SUTTON e BARTO, 1998), definido pelo seguinte processo iterativo:

$$v_{k+1}(i) = \min_{a \in A}\left\{r(i, a) + \sum_{j \in E} \lambda p_{ij}(a) v_k(j)\right\}, \tag{2.1}$$

onde prova-se que $v_k = (v_k^*(i); i \in E)$ converge para $v^* = (v^*(i); i \in E)$ a medida que

$k \to \infty$. O algoritmo da Iteração de Valor combina, em cada um de seus laços, uma avaliação da política atual de um passo e um aprimoramento desta política. A partir de uma constante θ é verificado o aprimoramento feito na política. Quando o valor desejado é atingido o algoritmo usará os menores valores $v(i)$, para cada estado i, para construir uma política otimizada π^* através da seguinte operação:

$$\pi^*(i) \leftarrow \text{argmin}_{a \in A} \left\{ r(i,a) + \sum_{j \in E} \lambda p_{ij}(a) v^*(j) \right\},$$

O Algoritmo 1 mostra o funcionamento do algoritmo da Iteração de Valor.

Algoritmo 1: Algoritmo de Iteração de Valor

```
1  Inicializa V(i) arbitrariamente;
2  repeat
3  |  Δ ← 0;
4  |  foreach i ∈ E do
5  |  |  v ← V(i);
6  |  |  V(i) = min_{a∈A}{r(i,a) + Σ_{j∈E} λ p_ij(a) v_k(j)};
7  |  |  Δ ← max{Δ, |v − V(i)|};
8  |  end foreach
9  until Δ < θ;
10 foreach i ∈ E do
11 |  π*(i) ← argmin_{a∈A}{r(i,a) + Σ_{j∈E} λ p_ij(a) v*(j)}
12 end foreach
```

2.1.2 Q-learning

Métodos de aprendizado por reforço são usados quando as probabilidades de transição $p_{ij}(a)$ são difíceis de serem determinadas. Neste trabalho foi utilizado o método Q-learning, introduzido por WATKINS C. (1989). Este método baseia-se na aproximação do processo iterativo dado pela Eq. (2.1), mas onde as probabilidade de transição são estimadas durante a execução do sistema ou através de uma simulação. Para introduzir esse método, é definido a seguinte função $Q_k(i,a)$ (matriz Q) para cada $i \in E$ e $a \in A$:

$$Q_{k+1}(i,a) \;\; := \;\; r(i,a) + \sum_{i \in E} \lambda p_{ij}(a) v_k(j) = \mathbb{E}_i \left[r(i,a) + \lambda v_k(X_1) \right] \tag{2.2}$$

onde v_k vem do processo iterativo dado pela Eq. (2.1). Foi utilizado X_1 arbitrariamente, sabendo que utilizando qualquer valor amostrado por X terá o mesmo resultado. Nota-se que $v_{k+1}(i) = \min_{a \in A} Q_{k+1}(i, a)$. A ideia é estimar a esperança do lado direito da Eq. (2.2) através da interação direta com o sistema a medida que ele evolui.

Para ilustrar como é feita essa estimativa, suponha que Y é uma variável aleatória. Suponha ainda que foram amostrados k valores para Y através de uma simulação, em que denota-se estes valores por y_1, y_2, ..., y_k. Pode-se estimar $\mathbb{E}f(Y)$, onde f é uma função real qualquer, da seguinte forma:

$$\mathbb{E}f(Y) \approx M_k := \frac{f(y_1) + f(y_2) + \cdots + f(y_k)}{k},$$

para k "grande". Nota-se ainda que se M_k foi calculado como acima, pode-se calcular M_{k+1} iterativamente após mais uma observação da variável aleatória Y. Isto é feito da seguinte forma:

$$M_{k+1} = \frac{k}{k+1} M_k + \frac{1}{k+1} f(y_{k+1}) = (1 - \alpha_k) M_k + \alpha_k f(y_{k+1})$$

onde y_{k+1} representa o resultado da nova amostragem e $\alpha_k = 1/(k+1)$.

Neste sentido, aplicando essa observação na definição de $Q_k(i, a)$ acima, é possível aproximar:

$$Q_{k+1}(i, a) = \mathbb{E}\left[r(i, a) + \lambda v_k(X_1)\right] \approx (1 - \alpha_k) Q_k(i, a) + \alpha_k (r(i, a) + \lambda v_k(j)),$$

onde j é o estado futuro quando saindo do par estado-ação (i, a). Além disso, usando o fato que $v_k = \min_{a \in A} Q_k(i, a)$ tem-se a fórmula de correção do algoritmo Q-learning:

$$Q_{k+1}(i, a) \leftarrow (1 - \alpha_k) Q_k(i, a) + \alpha_k (r(i, a) + \lambda \min_{a \in A} Q_k(j, a)),$$

onde geralmente chama-se α_k de taxa de aprendizado e nem sempre usa-se $\alpha_k = 1/(k+1)$. Essa correção é aplicada a cada transição de estado do sistema. Ao final de uma longa simulação, com, por exemplo, K transições, estima-se o controle ótimo por meio de π_K

que é dado por:

$$\pi_K(i) \leftarrow \text{argmin}_{a \in A} Q_K(i, a)$$

para todo $i \in E$. O Algoritmo 2 mostra o funcionamento do Q-learning.

Algoritmo 2: Algoritmo do Q-learning

Data: $Q(i, a)$ inicializado arbitrariamente
Inicializa i;
repeat
  Escolhe a de i utilizando a política derivada de Q;
  Executa ação a, observa $r(i, a)$ e s';
  $Q(i, a) \leftarrow (1 - \alpha)Q(i, a) + \alpha(r(i, a) + \lambda \min_{a \in A} Q_k(j, a))$;
  $s \leftarrow s'$;
until *i ser terminal*;

2.1.3 Processo de Decisão semi-Markoviano(SMDP)

Na subseção anterior foi apresentado o algoritmo Q-leaning para processos que evoluem a tempo discreto. Nesta seção será introduzido a adaptação do método Q-leaning, introduzido por BRADTKE (1994), para processos de decisão a tempo contínuo. Considere um processo $\{X_t\}$ tomando valor em um certo conjunto E que representa o estado de um sistema de interesse a cada instante $t \geq 0$, onde se supõe que este processo é homogêneo no tempo. Em certos instantes de tempo, chamados de épocas de decisão, um tomador de decisão pode escolher uma ação dentre um conjunto de possíveis ações A para tomar. Estes instantes de tomada de decisão são dados por uma sequencia de variáveis aleatórias $\{\sigma_n\}$, em que as diferenças $(\sigma_{n+1} - \sigma_n)$ são dadas pela variável aleatória τ_n, que são identicamente distribuídas. Deseja-se escolher uma política $\pi : E \to A$ que minimiza o custo do sistema, dado por:

$$w^{\pi}(i) = \mathbb{E}_i^{\pi}\left[\sum_{n=0}^{\infty} \int_{\sigma_n}^{\sigma_{n+1}} e^{-\beta t} c(X_t, \pi(X_n)) dt\right]$$

onde X_n é o estado do sistema no instante σ_n, $c(\cdot, \cdot)$ é uma "taxa" de custo do sistema e $\beta > 0$ é um fator de desconto. É possível mostrar sob certas condições (M. L. Puterman,

1994, pg. 541) que w^π satisfaz a seguinte equação, para qualquer política π:

$$w^\pi(i) = r(i, \pi(i)) + \mathbb{E}_i^\pi \left[e^{-\beta\tau_1} w^\pi(X_1) \right],$$

onde $r(i, a)$ é dado por

$$r(i, a) = \mathbb{E}_i^a \left[\int_0^{\tau_1} e^{-\beta t} c(X_t, a) dt \right] \tag{2.3}$$

τ_1 é uma variável aleatória representando a duração do tempo entre as duas épocas de decisão consecutivas, X_t representa o estado do sistema no tempo t e $\mathbb{E}_i^a$ representa a esperança condicionada à condição inicial i e o uso da ação a.

Outro resultado impontante, similar ao obtido para o caso a tempo discreto, é que pode-se encontrar o custo ótimo v^* através do seguinte processo iterativo:

$$v_{k+1}(i) = \max_{a \in A} \left\{ r(i, a) + \mathbb{E}_i^a \left[e^{-\beta\tau_1} v_k(X_1) \right] \right\},$$

em que X_1, representa o estado do sistema no fim de um intervalo de decisão. Desta forma, seguindo ideias análogas as desenvolvidas na seção anterior, tem-se a seguinte regra de atualização do algoritmo Q-learning, adaptado para processos de decisão a tempo contínuo:

$$Q_{k+1}(i, a) \leftarrow (1 - \alpha_k) Q_k(i, a) + \alpha_k \left[\tilde{r}(i, a) + e^{-\beta t} \min_{a \in A} Q_k(j, a) \right], \tag{2.4}$$

onde j e t são resultados da amostragem das variáveis aleatórias X_1 e τ_1 após a visita do par estado-ação (i, a), gerado pela simulação. Nota-se que $r(i, a)$ foi substituído por $\tilde{r}(i, a)$, que é uma aproximação de $r(i, a)$, já que é inviável determinar a integral exata dada por (2.3) durante a operação do sistema. Para ter uma estimativa da integral, o sistema é observado durante os instantes de tempos d_n, que ocorrem entre as épocas de decisão. Desta forma, podemos escrever:

$$\int_0^{\tau_1} e^{-\beta\tau} c(X_\tau, a) d\tau = \sum_{n=1}^{M} \int_{d_{n-1}}^{d_n} e^{-\beta\tau} c(X_\tau, a) d\tau,$$

em que M é o número de pontos de interrupção entre épocas de decisão consecutivas e $d_0 = 0$ e $d_M = \tau_1$. Desta forma, usando o teorema do valor médio integral, tem-se:

$$\int_{d_{n-1}}^{d_n} e^{-\beta\tau} c(X_\tau, a) d\tau \approx c(X_{d_n}, a) \int_{d_{n-1}}^{d_n} e^{-\beta\tau} d\tau = c(X_{d_n}, a) \frac{e^{-\beta d_{n-1}} - e^{-\beta d_n}}{\beta}$$

Logo, define-se $\tilde{r}(i, a)$ como sendo:

$$\tilde{r}(i, a) = \sum_{n=1}^{M} c(X_{d_n}, a) \frac{e^{-\beta d_{n-1}} - e^{-\beta d_n}}{\beta} \tag{2.5}$$

em que as variáveis aleatórias aparecendo na equação são substituídas pelos valores amostrados durante a simulação

3 Simulador

Neste capítulo será apresentado como foi implementada a simulação do *server farm* e a técnica usada na sua implementação. Esta simulação foi utilizada para testar o funcionamento do algoritmo Q-learning no sistema.

3.1 Simulação a Eventos Discretos

A Simulação a Eventos Discretos é caracterizada pela utilização de uma fila de eventos que acontecem a tempos descontínuos, sendo que o tempo da simulação não corresponde a sua duração real. Cada evento da simulação corresponde a um tempo t, que indica quando o evento ocorrerá. Estes eventos são organizados em uma fila, onde a ordenação é baseada nestes tempos dos eventos. Sejam os eventos e_a e e_b, que ocorrem nos tempos t_a e t_b respectivamente, onde $t_a < t_b$, se não existe eventos ocorrendo entre t_a e t_b o tempo da simulação pula de t_a diretamente para t_b.

3.2 Modelo do sistema

O modelo de *server farm* utilizado neste trabalho é composto de uma fila de clientes e um banco de servidores. O banco de servidores contém N blocos de servidores, sendo um bloco regular e $R = N - 1$ blocos chamados de reservas. Os blocos reservas contém necessariamente um número igual de servidores e estes blocos podem ser ligados e desligados conforme o controle utilizado. O número de servidores no bloco regular não é necessariamente igual ao número de servidores de um bloco reserva. Os servidores do bloco regular estão sempre ligados. Cada servidor consegue servir somente uma tarefa. A fila respeita o momento em que as tarefas chegam no sistema, a primeira a chegar é a primeira a ser servida (First In First Served - FIFS). Os servidores podem ser definidos como servidores livres, aqueles que podem ser usados para servir uma tarefa, servidores ocupados, aqueles que estão processando uma tarefa e servidores em espera, aqueles que

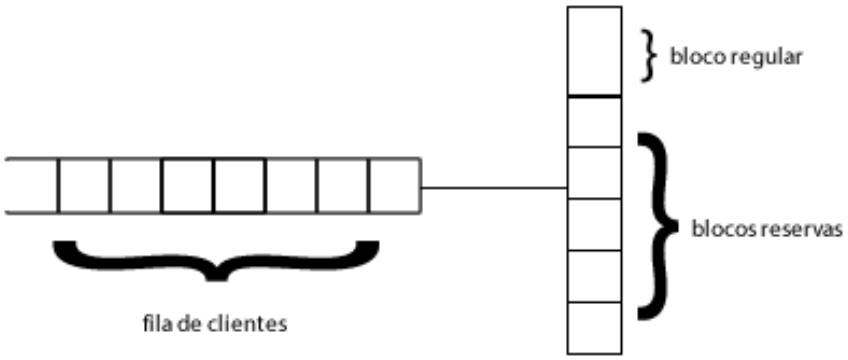

Figura 3.1: Representação do *server farm*

estão sendo ligados ou desligados consumindo energia sem poder atuar no processamento de tarefas. A Figura 3.1 ilustra o modelo do *server farm*.

O estado do sistema é composto por dois elementos, o número de clientes na fila e o número de blocos reservas ligados. A matriz Q é composta por três dimensões. Duas representam o estado do sistema (número de clientes e quantidade de blocos reservas ligados) e a outra representa o próxima quantidade de blocos reservas ligados. A Figura 3.2 mostra a definição da matriz Q.

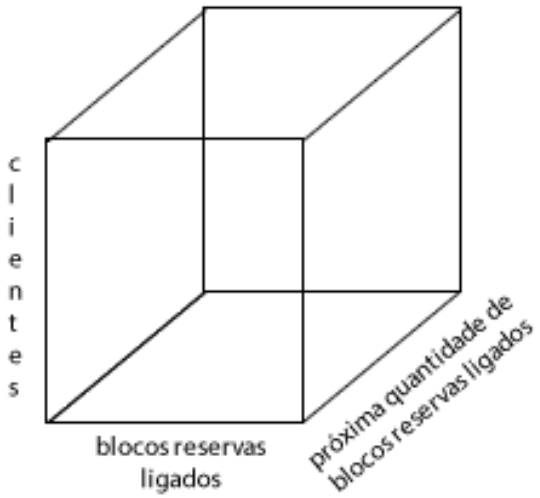

Figura 3.2: Representação da matriz Q

3.3 Implementação

A simulação de eventos discretos foi implementada na linguagem de programação C. Os eventos são gerados durante a simulação de forma consecutiva. Os tempos dos eventos podem seguir uma função de probabilidade exponencial ou hiper exponencial. Foram definidos oito tipos de eventos para a simulação. Eles são descritos a seguir:

- **Entrada de cliente (tarefa no sistema)**: corresponde à entrada de uma nova requisição no sistema. Ela é processada se existem servidores livres, caso contrário, a nova requisição fica na fila.
- **Saída de cliente**: acontece quando uma requisição é totalmente processada.
- **Ligar bloco de servidores reservas**: define o ligamento de um ou mais blocos de servidores reservas. Ele acontece após um sinal ser enviado ao sistema e possui um tempo de espera baseado no *setup time*.
- **Desligar bloco de servidores reservas**: define o desligamento de um ou mais blocos de servidores reservas. Ele acontece após um sinal ser enviado ao sistema e possui um tempo de espera baseado no *setup time*.
- **Tomada de decisão de ação**: corresponde a uma nova decisão do controle do sistema. Existem três ações possíveis: Ligar blocos de servidores reservas, desligar blocos de servidores reservas, manter a quantidade de blocos de servidores reservas.
- **Acumulo de valor**: define a soma dos custos atuais do sistema. Neste evento é calculado o valor de $\tilde{r}(i, a)$, dado pela Eq. (2.5)
- **Atualização da matriz Q**: define a atualização da matriz Q com o valor acumulado, usando a Eq. (2.4)
- **Cálculo de estatísticas**: define o cálculo dos dados provisórios do sistema utilizados para análise de resultados.

A simulação acontece em dois momentos diferentes. No primeiro momento, chamado de controle aleatório, as escolhas de ação são feitas aleatoriamente. Este primeiro

momento é importante para o preenchimento inicial da matriz Q. As decisões aleatórias proporcionam uma exploração maior dos estados e ações do sistema. Neste primeiro momento não é calculado nenhuma estatística. A duração do primeiro momento é uma porcentagem do tempo total de simulação. No segundo momento, chamado de controle guloso, as escolhas de ação são feitas seguindo a matriz Q, respeitando sempre a melhor ação naquele estado. A partir do início do segundo momento é gerado um evento de cálculo de estatística.

O sistema é inicializado com uma entrada de cliente. A cada tratamento de entrada de cliente é gerado um novo evento de entrada de cliente, representando o próximo cliente a entrar no sistema. A primeira tomada de ação acontece em um tempo determinado, as demais acontecem após a ação ser avaliada por um evento de atualização da matriz Q.

O sistema pode receber três tipos de sinais a cada decisão de ação. Um sinal para ligar blocos de servidores reservas, outro para desligar blocos e outro para manter o número de blocos ligados. Apesar do sinal ser uma ação atômica, o ligamento e desligamento dependem do *setup time* dos servidores. Para desligar blocos, o sistema deve ainda verificar a disponibilidade de servidores livres. Caso não tenha o número necessário no momento do sinal, o sistema verifica a cada evento de saída de cliente se é possível desligar mais blocos até que o número inicialmente desejado seja alcançado. Só então um novo evento de atualização da matriz Q é gerado.

Os eventos de acumulo de valor acontecem paralelamente ao funcionamento do sistema. Eles são importantes para que ao final de cada tomada de decisão, o valor utilizado para atualizar a matriz Q venha de uma média iterativa dos custos gerados durante esta ação. Ao final dos eventos de atualização da matriz Q é gerado um novo evento de tomada de decisão.

Os eventos de cálculo de estatísticas acontecem paralelamente ao funcionamento do sistema. Neles é calculado a média de clientes no sistema (clientes sendo servidos e clientes na fila), a quantidade total de servidores ligados (servidores livres e servidores em espera), a variância do número de clientes no sistema e a variância da quantidade total de servidores ligados. Estes dados são utilizados para a análise de resultados nos

capítulos seguintes. A simulação termina quando é tratado um evento com o tempo igual ou superior ao tempo de simulação pré definido.

A Figura 3.3 mostra o funcionamento da fila de eventos e do sistema de controle da simulação. O sistema se encontra com um servidor livre, e o primeiro evento é a da entrada do cliente "a". No mesmo momento este servidor começa a processar a requisição do cliente "a". Mais adiante, com a entrada do cliente "b", o sistema já lotado envia um sinal para ligar um bloco de servidores reservas. Durante o ligamento do bloco o processamento do cliente "a" é finalizado e a requisição do cliente "b" começa a ser processada. Após um tempo acontece a entrada do cliente "c" que só terá sua requisição processada com a finalização do ligamento do bloco (término do *setup time*).

Figura 3.3: Ilustração do funcionamento de uma fila de eventos hipotética e do sistema de controle da simulação

4 Resultados das Simulações

Neste capítulo será explicado os dados utilizados para fazer os testes e a analise dos resultados obtidos a partir da simulação implementada. Procura-se verificar e comparar as diferenças de custo, número de clientes no sistema e número de servidores utilizados em *server farm* onde não existe controle e *server farm* onde existe o controle feito com o algoritmo Q-learning utilizando um, quatro e oito blocos de servidores reservas. Todas as configurações de *server farm* tem o mesmo número total de servidores.

4.1 Base de Testes

Foram usados as seguintes configurações para a simulação:

- **Taxa de entrada de cliente**: 115
- **Taxa de serviço**: 1
- **Taxa de ligamento/desligamento de servidor**: 0,1
- **Tempo de simulação**: 2000000 unidades de tempo
- **Tempo de controle aleatório**: 50% do tempo de simulação
- **Coeficiente de variação da distribuição hiper exponencial**:2

Estas taxas são usadas para gerar os tempos dos eventos de entrada de cliente, saída de cliente, ligamento e desligamento de bloco de servidor extra. A Tabela 4.1 contém os dados dos números de servidores e blocos de servidores utilizados em cada simulação. Foram duas etapas de testes. A primeira etapa utilizou uma distribuição exponencial para o tempo de entrada de clientes e o tempo de processamento de tarefa. A segunda etapa utilizou uma distribuição hiper exponencial para o tempo de entrada de clientes e o tempo de processamento de tarefa. Para cada etapa foram feitas dez simulações com cada quantidade de blocos reservas, destes testes foram calculados a média e variância da

quantidade de servidores ligados, média e variância da quantidade de clientes no sistema e custo do sistema. O custo utilizado no treinamento do algoritmo Q-learning foi calculado a partir da seguinte formula:

$$C_1*(\textit{média de servidores ligados}) + C_2*(\textit{média de clientes no sistema})$$

Note que o custo combina objetivos conflitantes, que é o de manter poucos servidores ligados e manter poucas tarefas pendentes no sistema. Para estes testes, escolheu-se $C_1 = 1$ e $C_2 = 2$.

Tabela 4.1: Número de servidores e blocos reservas em cada base de teste

	Servidores regulares	Servidores reservas por bloco
1 bloco reserva	120	80
4 blocos reservas	120	20
8 blocos reservas	120	10

4.2 Análise dos Resultados

A Tabela 4.2 contém os resultados da média de servidores ligados durante as dez simulações para cada configuração. Estes dados mostram como o sistema se manteve durante a simulação. A Tabela 4.3 contém os resultados da variância da quantidade de servidores ligados para cada configuração, calculado durante a simulação. Estes dados mostram como o controle se comportou durante toda a simulação. A Tabela 4.4 contém os resultados da média de clientes no sistema. Estes dados incluem clientes nas fila e clientes com tarefas sendo processadas. A Tabela 4.5 contém os resultados da variância da quantidade de clientes no sistema, calculado durante a simulação. Estes dados mostram a frequência com que os clientes entraram no sistema. A Tabela 4.6 contém os resultados da média dos custos para as simulações. Estes dados são utilizados para verificar a eficácia do sistema de controle. Foi utilizado um intervalo de confiança de 95% para os resultados obtidos. Este intervalo é calculado utilizando a formula:

$$X \pm t_{n-1;\alpha/2}\frac{\sigma}{\sqrt{n}},$$

onde X é o valor obtido, $t_{n-1;\alpha/2}$ é valor corresponde a distribuição *t-student*, utilizada para calcular o intervalo de confiança, α é a constante correspondente a precisão do intervalo de confiança e n é o número de testes utilizados. Para este trabalho foi utilizado $\alpha = 0.025$ e $n = 10$ o que corresponde a $t_{n-1;\alpha/2} = 2,262$ e 95% de intervalo de confiança. O cálculo deste intervalo pode ser encontrado em (TRIVEDI K. S., 2002). Os dados nas tabelas estão escritos da seguinte forma:

(valor representado na tabela) $\pm$ *(intervalo de confiança)*

As Figuras 4.1, 4.2 e 4.3 ilustram a evolução da quantidade de servidores ligados pelo tempo da simulação para o controle com um, quatro e oito blocos reservas, respectivamente. Para cada uma das figuras, o gráfico posicionado na parte de cima representam os testes com distribuição exponencial e o de baixo com distribuição hiper exponencial. O eixo horizontal representa o tempo da simulação, e o exito vertical representa o número de servidores ligados. Nestes gráficos é possível perceber o impacto da variância da entrada de clientes no sistema de controle. Nos testes com distribuição exponencial existem mais épocas onde a quantidade de blocos reservas ligados é mantida. Nos casos com distribuição hiper exponencial a variação do número de blocos reservas ligados é maior.

Tabela 4.2: Resultados dos testes - média de servidores ligados

Distribuição exponencial	Média de servidores ligados
Sem controle	200 ± 0
1 bloco	$159,9648 \pm 0,2207$
4 blocos	$159,3062 \pm 0,2822$
8 blocos	$159,4212 \pm 0,0989$
Distribuição hiper exponencial	**Média de servidores ligados**
Sem controle	200 ± 0
1 bloco	$159,996 \pm 0,3254$
4 blocos	$159,0662 \pm 0,1387$
8 blocos	$159,1918 \pm 0,1027$

Tabela 4.3: Resultados dos testes - variância de servidores ligados

Distribuição exponencial	Variância de servidores ligados
1 bloco	$1600,0730 \pm 0,0619$
4 blocos	$787,4535 \pm 4,818$
8 blocos	$665,1956 \pm 3,312$
Distribuição hiper exponencial	Variância de servidores ligados
1 bloco	$1599,9737 \pm 0,1748$
4 blocos	$778,464 \pm 5,4172$
8 blocos	$663,0705 \pm 2,0807$

Tabela 4.4: Resultados dos testes - média de clientes no sistema

Distribuição exponencial	Média de clientes no sistema
Sem controle	$115,0461 \pm 0,0719$
1 bloco	$122,7081 \pm 0,1241$
4 blocos	$118,6033 \pm 0,1396$
8 blocos	$117,2515 \pm 0,0902$
Distribuição hiper exponencial	Média de clientes no sistema
Sem controle	$114,9962 \pm 0,1195$
1 bloco	$128,8967 \pm 0,2781$
4 blocos	$121,4808 \pm 0,1321$
8 blocos	$119,1820 \pm 0,2036$

Tabela 4.5: Resultados dos testes - variância de clientes no sistema

Distribuição exponencial	Variância de clientes no sistema
Sem controle	$115,1129 \pm 1,5980$
1 bloco	$410,7736 \pm 6,3044$
4 blocos	$262,8215 \pm 5,3109$
8 blocos	$207,0642 \pm 5,0137$
Distribuição hiper exponencial	Variância de clientes no sistema
Sem controle	$162,6403 \pm 1,8439$
1 bloco	$1052,3070 \pm 34,1023$
4 blocos	$593,3458 \pm 19,9383$
8 blocos	$446,737 \pm 16,4191$

Tabela 4.6: Resultados dos testes - média do custo do sistema

Distribuição exponencial	Média do custo do sistema
Sem controle	$430,0921 \pm 0,1438$
1 bloco	$405,3810 \pm 0,3321$
4 blocos	$396,5128 \pm 0,3097$
8 blocos	$393,9242 \pm 0,1588$
Distribuição hiper exponencial	Média do custo do sistema
Sem controle	$429,9923 \pm 0,2387$
1 bloco	$417,7894 \pm 0,5924$
4 blocos	$402,0279 \pm 0,2251$
8 blocos	$397,5559 \pm 0,4063$

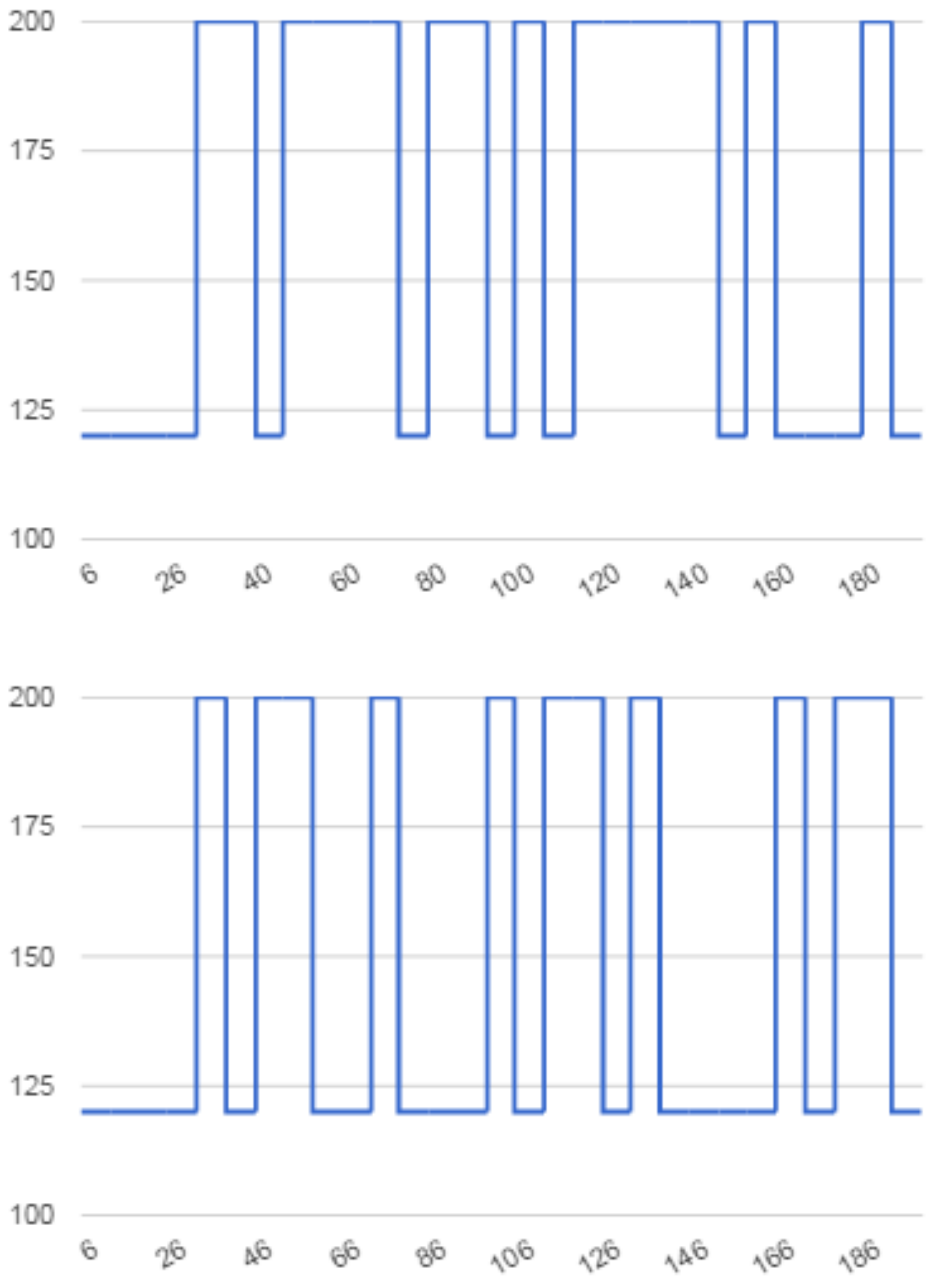

Figura 4.1: Evolução do número de servidores ligados para o sistema de controle com um bloco reserva

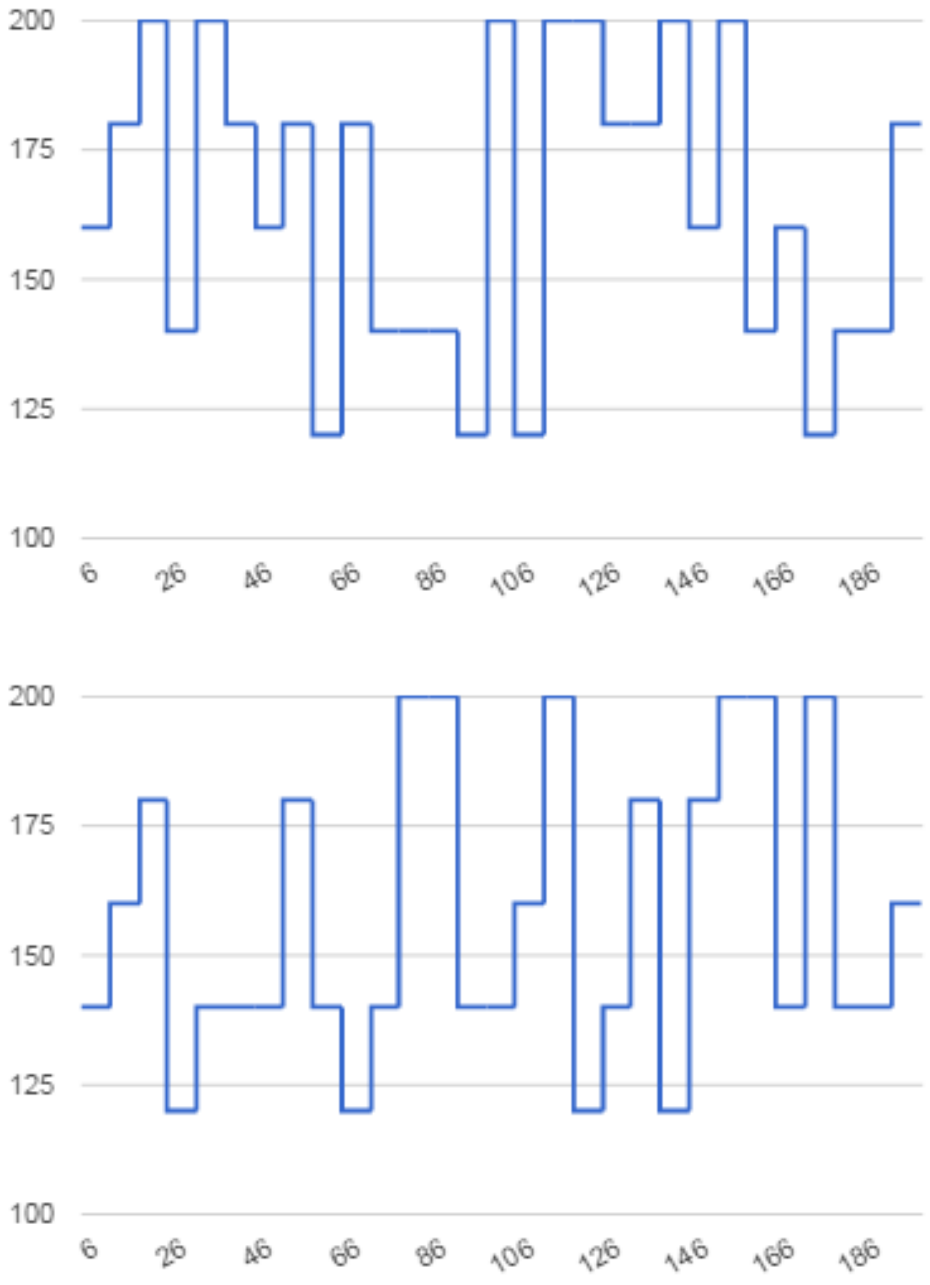

Figura 4.2: Evolução do número de servidores ligados para o sistema de controle com quatro blocos reservas

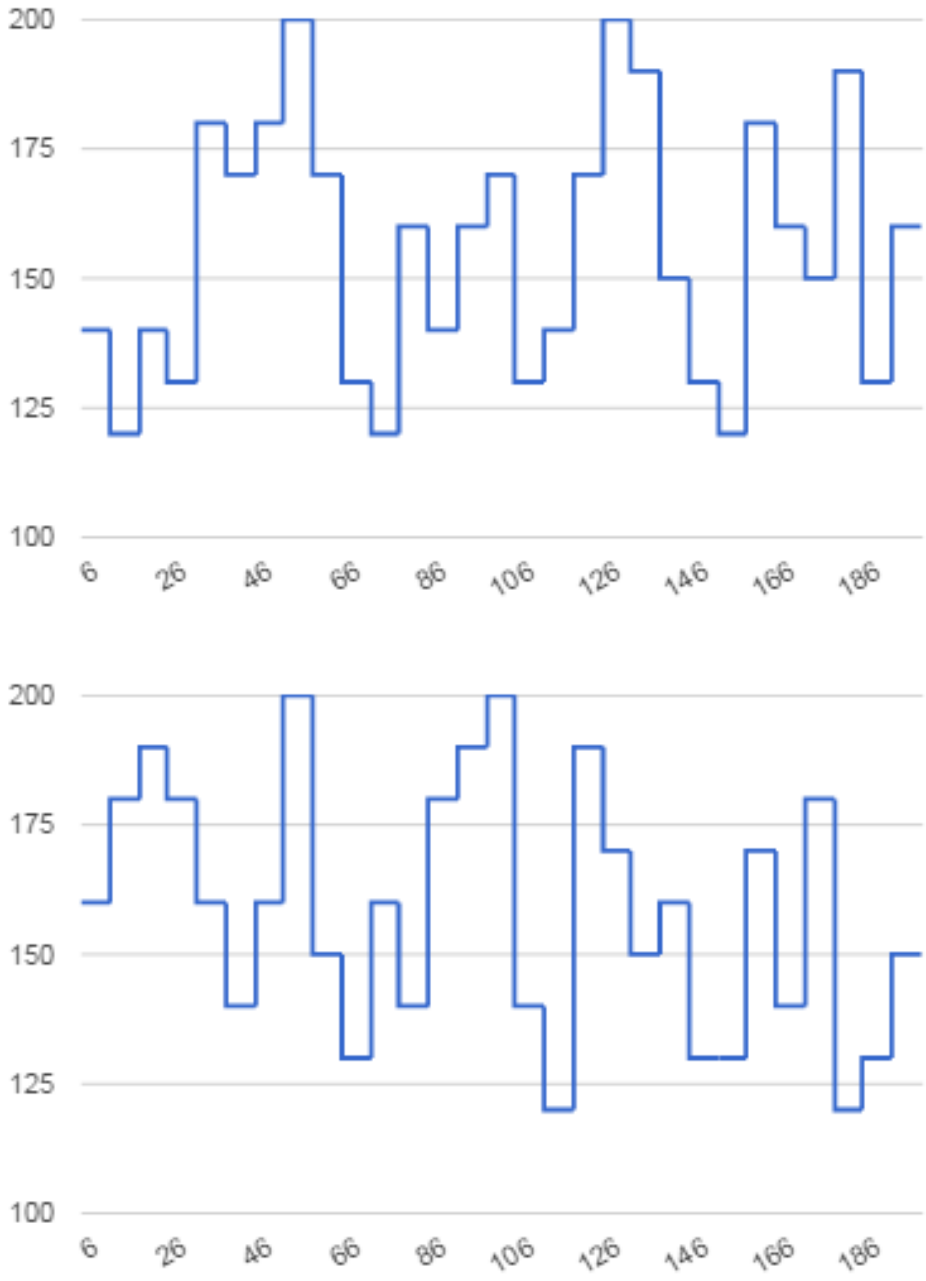

Figura 4.3: Evolução do número de servidores ligados para o sistema de controle com oito blocos reservas

5 Conclusões

Neste trabalho foi considerado a utilização de um método de aprendizado por reforço para fazer o controle de *server farms* visando diminuir o custo com o gasto de energia mantendo a qualidade do serviço. A simulação a eventos discretos conseguiu reproduzir o comportamento de um *server farm*. O algoritmo Q-learning encontrou uma política de controle que minimizou o custo do sistema tanto para a proposta de entrada com distribuição exponencial como a proposta de distribuição hiper exponencial. A importância da utilização das duas hipóteses de distribuição para entrada e serviço no sistema se da pelo fato de que os sistemas reais não trabalham com uma demanda constante, existem épocas com demandas variadas. A utilização de um modelo de *server farm* com vários blocos de servidores reservas foi importante para melhor controlar a variação da demanda em diferentes épocas. Quanto maior a divisão do número de servidores maior é o alcance do controle, podendo assim atingir mais níveis de lotação do sistema, mantendo um estado estável quanto ao número de clientes no sistema.

Referências Bibliográficas

Artalejo, J. R.; Economou, A. ; Lopez-Herrero, M. J. Analysis of a multiserver queue with setup times. **Queueing Systems**, v.51, n.1, p. 53–76, 2005.

Bradtke, S. J.; Duff, M. O. **Reinforcement learning methods for continuous-time markov decision problems**. In: Advances in Neural Information Processing Systems, p. 393–400. MIT Press, 1994.

Le Sueur, E.; Heiser, G. **Dynamic voltage and frequency scaling: The laws of diminishing returns**. In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, HotPower'10, p. 1–8, Berkeley, CA, USA, 2010. USENIX Association.

Gandhi, A.; Gupta, V.; Harchol-Balter, M. ; Kozuch, M. A. Optimality analysis of energy-performance trade-off for server farm management. **Perform. Eval.**, v.67, n.11, p. 1155–1171, Nov. 2010.

Greenberg, A.; Hamilton, J.; Maltz, D. A. ; Patel, P. The cost of a cloud: Research problems in data center networks. **SIGCOMM Comput. Commun. Rev.**, v.39, n.1, p. 68–73, Dez. 2008.

Lee, N.; Kulkarni, V. G. Optimal arrival rate and service rate control of multi-server queues. **Queueing Systems**, v.76, n.1, p. 37–50, 2014.

Puterman, M. L. **Markov Decision Processes: Discrete Stochastic Dynamic Programming**. 1st. ed., New York, NY, USA: John Wiley & Sons, Inc., 1994.

Mazzucco, M.; Dyachuk, D. Optimizing cloud providers revenues via energy efficient server allocation. **Sustainable Computing: Informatics and Systems**, v.2, n.1, p. 1 – 12, 2012.

Marra, J. G. S.; Leite, S. C. Controle do consumo de energia em data centers via aprendizado por reforço. **Anais do CILAMCE 2013**, p. 125, Nov 2013.

Mitrani, I. **Computer Performance Engineering: 9th European Workshop, EPEW 2012, Munich, Germany, July 30, 2012, and 28th UK Workshop, UKPEW 2012, Edinburgh, UK, July 2, 2012**, chapter Trading Power Consumption against Performance by Reserving Blocks of Servers, p. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Council, N. R. D. **America's data centers are wasting huge amounts of energy**. Disponível em http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IB.pdf, 2014. Acesso em 08 de Março de 2016.

Niyato, D.; Chaisiri, S. ; Sung, L. B. **Optimal power management for server farm to support green computing**. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, p. 84–91, Washington, DC, USA, 2009. IEEE Computer Society.

Sutton, R. S.; Barto, A. G. **Introduction to Reinforcement Learning**. 1st. ed., Cambridge, MA, USA: MIT Press, 1998.

Trivedi, K. S. **Probability and Statistics with Reliability, Queuing and Computer Science Applications**. 2nd edition. ed., Chichester, UK: John Wiley and Sons Ltd., 2002.

União internacional das telecomunicações. Disponível em http://www.itu.int/en/Pages/default.aspx, 2014. Acesso em 23 de Fevereiro de 2016.

Watkins, C. **Learning from Delayed Rewards**. 1989. Tese de Doutorado - University of Cambridge,England.