Avaliando a Detecção e o Tratamento de *Flash Crowds* Utilizando o SimGrid

Vitor Pinheiro David[1], **Caio César A. Sampaio**[1], **Eduardo dos Santos Costa**[1], **Ubiratam de Paula** [1]

[1] Universidade Federal Rural do Rio de Janeiro (UFRRJ)

vitor.pito@gmail.com, caio.cesarsampaio@hotmail.com

eduardo00100@hotmail.com, upaula@ufrrj.br

Abstract. *This article proposes a simulation, where cloud resources can be dynamically hired in order to mitigate the effects of flash crowds events. This simulation was developed through the integration of SimGrid simulator with existing solutions in the literature for the flash crowds detection and handling problems. The initial tests were satisfactory and promising. For future works, we intend to perform more tests with larger scenarios.*

Resumo. *Este artigo propõe uma simulação, onde recursos da nuvem podem ser contratados dinamicamente com a finalidade de mitigar os efeitos dos eventos de flash crowds. Esta simulação foi desenvolvida através da integração do simulador SimGrid com soluções já existentes na literatura para a detecção e tratamento de flash crowds. Os testes iniciais foram satisfatórios e promissores. Futuramente, pretende-se realizar mais testes com cenários maiores.*

1. Introdução

Com o aumento da popularidade da Internet e, consequentemente, da quantidade de seus usuários, conteúdos podem se tornar virais muito rapidamente. Nestes casos, os conteúdos podem ficar indisponíveis ou com baixa qualidade de serviço (QoS). Estes eventos são denominados *flash crowd* e precisam ser detectados e tratados de maneira rápida e eficiente.

Existem muitas definições de *flash crowd* na literatura [Stavrou et al. 2002, Wendell and Freedman 2011, Atajanov et al. 2007]. Em [Stavrou et al. 2002], uma *flash crowd* é definida como um fenômeno resultante de um aumento repentino e imprevisível na popularidade de um conteúdo online. Em [Wendell and Freedman 2011], os autores definem *flash crowd* como um período no qual as taxas de requisições para um domínio crescem exponencialmente. Dessa forma, pode-se concluir que um evento de *flash crowd* acontece quando as taxas de acesso de um conteúdo ou domínio sofrem um aumento além do esperado para o tráfego normal. Por exemplo, um site de notícias possui um taxa média de acessos por dia. Entretanto, quando algum fato noticiado possui grande interesse ou popularidade, como ataques terroristas ou desastres naturais, as taxas de acesso sofrem um grande aumento, ultrapassando a taxa média normal esperada. Também é possível observar um cenário similar nos períodos de inscrições de disciplinas nos sites/sistemas das universidades, onde o fluxo de requisições aumenta consideravelmente.

Várias soluções para este problema já foram estudadas por diversos pesquisadores [Wendell and Freedman 2011, Sladescu et al. 2013, Atajanov et al. 2007,

de Paula et al. 2015, de Paula 2015]. Em [de Paula 2015], foi apresentado um mecanismo de detecção de *flash crowd*, chamado FCD (*Flash Crowd Detection*), que utiliza conceitos de Teoria da Informação, como entropia e correlação. A maior vantagem do FCD em relação à literatura relacionada é que não é necessário definir uma escala de crescimento para o número de acesso, ou seja, não é preciso tentar quantificar o aumento do número de acessos através de determinadas características, como crescimento exponencial. Além disso, é possível calcular os valores máximos para as entropias e correlação total utilizando apenas o número de conteúdos, que é conhecido.

Também em [de Paula 2015], o tratamento deste eventos foi modelado matematicamente como um problema de programação inteira, chamado *Flash Crowd Handling Problem - Integer Programming* (FCHP-IP). Neste modelo, foi incluído a possibilidade de contratação de recursos elásticos (Computação em Nuvem) para atender de forma satisfatória as demandas de requisições. Como o tempo para resolver as formulações matemáticas pode ser elevado (dependendo do tamanho dos dados de entrada), uma heurística, chamada FCHP-ILS, também foi proposta. A FCHP-ILS é baseada na metaheurística *Iterated Local Search* (ILS)[Talbi 2009, Lourenço et al. 2003] e por quatro componentes: (i) um procedimento para gerar uma solução inicial viável atribuindo cada requisição ao servidor com menor custo e com largura de banda disponível; (ii) uma busca local que tenta encontrar uma nova solução viável de menor custo a partir da solução atual; (iii) uma estratégia de perturbação que visa escapar de ótimos locais ainda distantes dos ótimos globais; e (iv) um critério de aceitação elitista para novas soluções, onde apenas soluções com menor custo são aceitas e utilizadas nas iterações seguintes. Por fim, ainda em [de Paula 2015], o mecanismo de detecção FCD e a heurística FCHP-ILS foram avaliados teoricamente e em pequenos cenários executados na nuvem da Amazon EC2 [Amazon Web Services 2020].

Devido ao alto custo monetário para avaliar estas soluções em ambientes reais, neste artigo é apresentada uma proposta de simulação para este problema utilizando o simulador SimGrid [Casanova et al. 2014]. Desta forma, pode-se avaliar a qualidade das soluções em cenários maiores e que representam situações reais de *flash crowds*.

O simulador SimGrid foi escolhido, mais especificadamente seu módulo *s4u*, pelas seguintes razões: (i) ser amplamente utilizado pela comunidade científica, fato comprovado pela grande quantidade de artigos publicados que o utilizam[1], (ii) ter atualizações recentes, atualmente, na versão 3.24[2], e (iii) ter fórum ativo de colaboradores[3]. Não foram encontrados trabalhos na literatura relacionada que utilizam simuladores de ambiente de nuvens de computadores para o problema abordado.

2. Simulação Realizada

Para avaliar as soluções propostas em [de Paula 2015], foi necessário construir uma simulação com a estrutura representada na Figura 1. Esta estrutura é baseada nos experimentos reais que foram realizados na nuvem da Amazon EC2. A diferença é que todos os servidores e usuários são implementados no simulador.

Na Figura 1, é possível observar que as requisições dos clientes são recebidas

[1] `https://simgrid.org/usages.html`
[2] `https://github.com/simgrid/simgrid`
[3] `https://lists.gforge.inria.fr/pipermail/simgrid-user/`

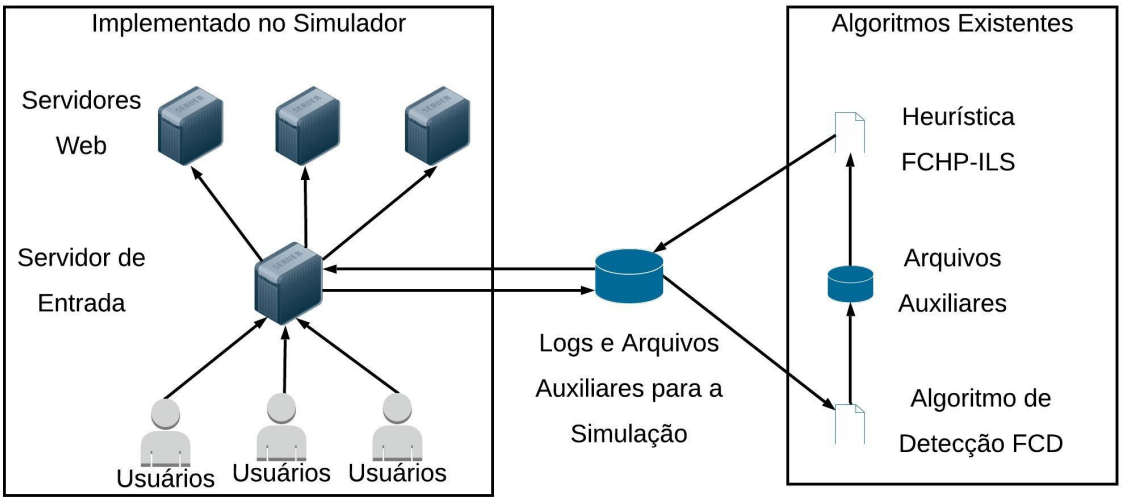

Figura 1. Proposta de simulação com o SimGrid para o problema.

por um servidor de entrada. Este, além de registrar as requisições no arquivo de *log*, também as redireciona para um servidor Web capaz de atendê-las. O mecanismo FCD é continuamente executado tendo como entrada o arquivo de *log*. Caso um início ou término de *flash crowd* seja detectado, a heurística FCHP-ILS é executada para verificar se a solução atual precisa ser modificada. Essa modificação pode um reposicionamento dos conteúdos nos servidores já existentes, uma contratação de novos servidores ou ambos. Este retorno para o simulador é feito através de leitura e escrita em arquivos.

A utilização de arquivos é uma solução mais elegante, porém a troca de dados no módulo *s4u* do SimGrid funciona através de *mailboxs*. As *mailboxs* realizam a comunicação entre atores que tenham uma rota de comunicação definida entre si. Os atores são criados a partir de *hosts*. O *host* é a estrutura de dados que define as características dos atores derivados dela. Além disso, essa estrutura é utilizada na criação das rotas de comunicação entre atores e define o que será representado, por exemplo, um servidor ou um usuário.

Atores são instâncias de *hosts*, ou seja, são as entidades que realmente executam durante a simulação. Suas características, como poder de processamento, número de núcleos, entre outros, são definidas pelo seu *host*. Cada ator tem uma função associada a ele e sempre que um ator é criado, essa função começa a executar e quando ela acaba o ator é excluído.

A simulação é executada a partir da criação de atores e da troca de mensagens entre eles. A criação dos atores é possível por código ou pela leitura de arquivos XML de entrada. As Figuras 2 e 3 ilustram exemplos destes arquivos. Na Figura 2, são definidos dois *hosts*, dois *links* e uma rota (composta pelos dois *links*). Na Figura 3, é definida qual função será executada pelo ator associado a um *host*. As informações necessárias para a criação destes arquivos são adquiridas de *traces* reais de *flash crowds*, que também foram utilizados em [de Paula 2015].

A contratação de recursos na nuvem e novas replicações são realizadas no simula-

```
<?xml version'1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">

<platform version="4.1">
  <zone id="first zone" routing="Full"><!-- Inicia uma rede -->
    <host id="C1" speed="1Mf"/> <!-- Inicia um host, nesse caso um cliente -->

    <host id="S1" speed="1Mf"/> <!-- Nesse caso um servidor -->

    <link id="C1" bandwidth="100Bps" latency="0ms"/> <!-- Inicia um link -->

    <link id="S1" bandwidth="1000MBps" latency="0ms"/>

    <route src="C1" dst="S1"> <!-- Inicia uma rota entre a origem e o destino -->
      <link_ctn id="C1"/> <!-- Referencia ao link já existente com id = C1 -->
      <link_ctn id="S1"/>
    </route>
  </zone>
</platform>
```

Figura 2. Exemplo de XML utilizado pelo SimGrid para definir a rede.

```
<?xml version'1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">

<platform version="4.1">

  <actor host="C1" function="request"> <!-- Inicia um ator com host = C1
  e esse ator executa a função request -->
    <argument value="exemplo"/> <!-- valor que é passado como parametro
    para a função do ator-->
  </actor>

</platform>
```

Figura 3. Exemplo de XML utilizado pelo SimGrid para iniciar os atores.

dor através das informações contidas em um arquivo auxiliar, gerado a partir do resultado da heurística FCHP-ILS. Os recursos elásticos são representados como servidores já definidos no arquivo XML, porém os atores que os representam só são criados de acordo com o arquivo auxiliar.

As simulações foram realizadas para reproduzir os dois cenários reais apresentados em [de Paula 2015], com *traces* de 1041 e 1798 requisições de um dia completo. Os resultados foram compatíveis e obtidos em menos de um segundo de simulação.

3. Conclusão e Trabalhos Futuros

Neste artigo foi apresentada uma proposta de simulação utilizando o SimGrid integrado às soluções para a detecção e tratamento de *flash crowds* propostas em [de Paula 2015]. Os resultados iniciais são promissores e compatíveis com os resultados encontrados nas execuções reais na nuvem da Amazon, considerando os mesmos *traces*. O desempenho da proposta também se mostrou satisfatório, apresentando tempo de simulação inferior a um segundo nos testes realizados. Como trabalhos futuros, espera-se realizar mais testes, principalmente em cenários reais e com diferentes tamanhos de eventos de *flash crowds*.

4. Agradecimentos

Os autores gostariam de agradecer à FAPERJ pela bolsa de iniciação científica concedida.

Referências

Amazon Web Services, I. (2020). Amazon EC2. `https://aws.amazon.com/pt/ec2/`.

Atajanov, M., Shimokawa, T., and Yoshida, N. (2007). Autonomic multi-server distribution in flash crowds alleviation network. In *Emerging Directions in Embedded and Ubiquitous Computing*, volume 4809 of *Lecture Notes in Computer Science*, pages 309–320. Springer Berlin Heidelberg.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

de Paula, U. (2015). *Detecção e Tratamento de Eventos de Flash Crowd em Nuvens Computacionais*. PhD thesis, Universidade Federal Fluminense - Instituto de Computação.

de Paula, U., Drummond, L., de Oliveira, D., Frota, Y., and Barbosa, V. C. (2015). Handling flash-crowd events to improve the performance of web applications. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 769–774, New York, NY, USA. ACM.

Lourenço, H., Martin, O., and Stützle, T. (2003). Iterated local search. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer US.

Sladescu, M., Fekete, A., Lee, K., and Liu, A. (2013). A polymorphic model for event associated workload bursts. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pages 119–125.

Stavrou, A., Rubenstein, D., and Sahu, S. (2002). A lightweight, robust p2p system to handle flash crowds. *SIGCOMM Comput. Commun. Rev.*, 32(3):17–17.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing.

Wendell, P. and Freedman, M. J. (2011). Going viral: flash crowds in an open CDN. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, IMC '11, pages 549–558, New York, NY, USA. ACM.