ANALISE DE VIABILIDADE DE USO DE RASPBERRY PI NA NÉVOA

GUILHERME DE SOUZA[1]; JOÃO SILVEIRA[2]; GIOVANI JAHN[3]; SAMUEL BESKOW[4]; GERSON GERALDO H. CAVALHEIRO[5]

[1]Universidade Federal de Pelotas – gdsdsilva@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas – jpdmdsilveira@inf.ufpel.edu.br
[3]Universidade Federal de Pelotas – giovani.jahn@iffarroupilha.edu.br
[4]Universidade Federal de Pelotas – samuel.beskow@ufpel.edu.br
[5]Universidade Federal de Pelotas – gerson.cavalheiro@inf.ufpel.edu.br

1. INTRODUÇÃO

Em virtude da explosão do chamado *big data*, decorrente do crescimento da oferta e emprego de dispositivos que produzem e/ou consomem dados, como dispositivos inteligentes, vestíveis e sistemas em veículos, as aplicações próprias ao processamento de dados alcançaram novos patamares de importância e são cada vez mais requisitadas pelos indivíduos. A primeira alternativa para viabilizar computacionalmente toda a carga de processamento gerada foi empregar os recursos disponíveis na nuvem. Como vantagem adicional, a nuvem também reúne características que permitem a agregação de dados de diversas fontes. No entanto, como consequência direta do uso da nuvem, observou-se tanto um grande aumento no tráfego de rede como dificuldades quando o acesso à rede, pelo usuário final, é limitado ou mesmo inexistente.

Considerando que o processamento pode ocorrer em *data centers* geograficamente distantes, aplicações que dependem de uma baixa latência tendem a sofrer com o caminho percorrido até a nuvem. Inúmeras dessas aplicações, oriundas da Internet das Coisas (*Internet of Things* – IoT) (GUBBI, et all, 2013), encontram-se nessa situação. Nesses casos, a qualidade da rede entre o ponto onde o dado é gerado ou necessário até o ponto onde existe o processamento impacta diretamente a qualidade do resultado obtido. Como mecanismo de compensação, data centers são forçados a se adequar, seja aumentando recursos ou fazendo um maior aproveitamento dos existentes. Muitas vezes, esse esforço de adequação resulta em um aumento no consumo energético de tais centros de computação, o qual, em 2012, aproximava-se de 270 TWh, crescendo anualmente (VAN HEDDEGHEM et all, 2014).

De forma a atender melhor essa demanda, um novo paradigma foi proposto, a *Computação em Névoa* (BONOMI et all, 2012), que estende a Computação em Nuvem (MELL et all, 2011), trazendo a possibilidade de se ter dispositivos habilitados ao processamento no meio do caminho. A computação em névoa traz, dessa forma, o processamento ou pré-processamento de dados para perto do usuário, mantendo seus quatro pontos essenciais: baixa latência, maior confiabilidade, peso reduzido de *hardware* (tamanho, local e número de máquinas) e baixo consumo energético (BYERS, 2017).

Para o processamento desses dados, nem sempre há como utilizar uma máquina com grande poder computacional e, muitas vezes, não se apresenta essa necessidade. Com isso, a névoa traz a possibilidade do uso de dispositivos de baixa capacidade que possam suprir as necessidades geradas pelas aplicações. Em SOUZA et all, 2018, foi proposto o uso de dispositivos com arquitetura ARM, em específico a Raspberry Pi 3, como parte de uma névoa, mantendo o foco em avaliar o consumo energético gerado por ela.

De forma semelhante, o trabalho aqui apresentado tem por objetivo avaliar e quantificar seu desempenho com relação à latência por meio de estudos práticos a partir da simulação de escrita e leitura em banco de dados NoSQL com o *benchmark* Netflix Data Store Benchmark (NDBench) (PAPAPANAGIOTOU et all, 2018). Este experimento nos permite criar cenários próximos à realidade de diversas aplicações dado o contexto de serviços web. Por exemplo: blogposts, *sites* de informações, *marketplaces*, além do processamento de dados oriundos da coleta de sensores, seja dentro da agroindútria ou em empresas autônomas, ou seja, é possivel pensar em inúmeras aplicações dada a quantidade de atendimento necessário.

Fazendo uso de diferentes tipos de carga, tornou-se possível encontrar resultados adequados próximos ao limite do dispositivo, obtendo-se assim resultados de latência nos percentis 95 e 99, latênca média e operações por segundo. Os resultados demonstraram que a Raspberry Pi 3 possui capacidade de atendimento de aplicações que se assemelham aos experimentos aqui realizados.

2. METODOLOGIA

Visto que a demanda de serviços web ocupa a maior parte das aplicações na nuvem, para a realização deste trabalho optou-se pelo uso de um banco de dados NoSQL, dada sua adequação ao uso em sistemas distribuídos e em aplicações com alta demanda e dependentes de elasticidade (CATTELL, 2011). Tendo tudo isso em vista, optamos por fazer uso do banco de dados NoSQL Apache Cassandra.

Da mesma forma, para escolha do simulador entre os *benchmarks* disponíveis para testes em sistemas de nuvem, optou-se pelo que seria capaz de atender nossas exigências de latência e que nos permitisse uma livre escolha das cargas de trabalho para criarmos cenários restritos para execução dos experimentos com cargas ideais e próximas à capacidade de processamento da Raspberry. Dadas tais necessidades, entre outras funções, o simulador NDBench foi o selecionado para o presente trabalho.

Visando cenários onde se aproximem da realidade de muitas aplicações disponiveis nos tuas atuais, iniciamos a preparação dos casos de estudo. Inicialmente, foi instalado na Raspberry Pi 3 o ambiente do Cassandra na sua versão 2.2.14, por questão de compatibilidade. Similarmente, configurou-se a máquina-cliente que manteve o NDBench ativo, para isso segui-se a documentação de ambas ferramentas.

A comunicação entre as máquinas foi realizada por canal criado pelas portas *ethernet*, via cabo de rede, de forma a diminuir o ruído nos resultados obtidos. A configuração da máquina cliente manteve as configurações *default*, ou seja, alguns valores pré selecionados pelo simulador foram mantidos. Somente opções como *numWrites/numReads* e *writeRateLimit/readRateLimit* foram alteradas, de forma a alcançar um *benchmark* no qual fosse possível chegar próximo a um limite do dispositivo e analisar um real cenário de aplicação.

Todos os experimentos contaram com a duração de 25 minutos, sem sofrer alteração na configuração do simulador durante a execução, embora o simulador nos permitisse isso. Experimentos foram executados anteriormente para analisar as possíveis variações, cargas e falhas dada pelo simulador, não sofrendo variações, empregou-se cargas aproximadas do limite de resposta da Raspberry

Pi 3 para a obtenção dos resultados. Com as cargas selecionadas, criaram-se três cenários, de forma a simular determinadas situações. Inicialmente realizamos um teste que simula somente escritas no banco, depois outro que simula somente leituras e, por fim, uma simulação onde ambos ocorrem, com 75% de leituras e 25% de escritas. Assim a configuração para execução dos três experimentos manteve 256 threads para *numWrite* e *numRead*. Mantendo *writeRateLimit* e *readRateLimit* em 2.000w/s e 0r/s, 0w/s e 2.000r/s por fim 500w/s e 1.500r/s.

3. RESULTADOS E DISCUSSÃO

A escolha em demonstrar os resultados não somente em latência média, mas também em percentil 95 e 99 vem das informações que a média nos esconde, sendo atualmente a métrica mais utilizada em servidores. Logo, com isso temos resultados como 95% dos usuários são atendidos abaixo de uma certa latência, ou seja, o servidor pode estar atendendo alguns usuários com um latência ruim e isso variará com a quantidade de usuários que o servidor esta atendendo (o mesmo ocorre ao utilizar o percentil 99).

O montante de operações realizadas durante os 25 minutos de execução do simulador, alcançou em torno de dois milhões de operações em ambos cenários. O Experimento 1 apresentou uma taxa de falhas em torno de ≈ 0.1298%, enquanto os Experimentos 2 e 3 apresentaram zero e 26 falhas respectivamente, caindo em *time out*, possivelmente pela demora da Raspberry Pi 3 em atender a demanda.

No primeiro cenário, no qual realizamos somente escritas, observou-se que a Raspberry atendia mais de 1.200 operações por segundo. Mantendo a latência média e a percentil 95 abaixo dos 500ms. Onde havia somente leituras, a *small board* se manteve entre 1.200 e 1.500 operações por segundo. Igualmente, manteve uma latência de qualidade, mantendo a média abaixo de 500ms e percentil 95 abaixo de 1000ms. Por fim, no ultimo experimento, demonstrou uma baixa latência, tanto em percentil 95 estando um pouco a mais de 500ms, quanto a média, que como nos outros testes se manteve abaixo dos 500ms. Ao olharmos para as operações que realizou durante o período de execução do simulador, podemos ver que chegava em torno de mais de 450 requisições por segundo de escritas e leituras. Visto que nos resultados anteriores os alcances em requisições foram maiores, pudemos observar que a placa tenta distribuir igualmente o trabalho de forma a atender ambas necessidades.

4. CONCLUSÕES

De acordo a seção anterior, a Raspberry Pi 3 mostrou-se eficiente nos cenários propostos, alcançando boas respostas em latência, dado que ao observar os resultados em percentil 95 se mantém entre 500ms e 1.000ms. Observando a média, em todas operações se mantiveram abaixo dos 500ms que, para inúmeras aplicações, torna-se mais que necessário e viabiliza o ajuste de muitas aplicações que podem vir a se enquadrar. Podemos ver que ela atende uma boa demanda de requisições por segundo, mantendo-se em torno de 1.100 operações, que representa uma quantidade de atendimento adequada a uma grande gama de aplicações.

Com tais dados, pudemos demonstrar a capacidade da Raspberry Pi 3 em atender a necessidades de muitas aplicações, suprindo os requisitos exigidos pelo paradigma de névoa, tendo em vista os quatro pontos essenciais da névoa.

O trabalho aqui apresentado quantificou um pouco do poder do dispositivo em resposta de latência para aplicações semelhantes aos cenários aqui propostos. Assim possibilitando estudos com aplicações reais com conjuntos complexos de calculos, dessa forma fazendo seu uso como principal/parte de processamento como servidores, ainda permitindo testes de escalabilidade tanto horizontal quanto vertical.

5. REFERÊNCIAS BIBLIOGRÁFICAS

GUBBI, J., BUYYA, R., MARUSIC, S., PALANISWAMI, M. Internet of Things (IoT): A vision, architctural elements, and future direction. **Future Generation Computer Systems**, 1645-1660, 2013.

VAN HEDDEGHEM, W., LAMBERT, S., LANNOO, B., COLLE, D., PICKAVET, M., DEMEESTER, P. Trends in wordlwide ict electricty consumpition from 2007 to 2012. **Comput. Commun.**, 64-76, 2014.

BONOMI, F., MILITO, R., ZHU, J., ADDEPALLI, S. Fog computing and its role in the internet of things. **In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing**. 13-16, 2012.

Mell, P., GRANCE, T. The NIST definition od cloud computing. **NIST Special Publication**. 2011.

BYERS, C. C. Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks. **IEEE Communications Magazine**. 14-20, 2017.

SOUZA, G., OLIVEIRA, J., PILLA, M. Comparação de desempenho do workload ycsb em raspberry pi b+ e 3. **In XVIII Escola Regional de Alto Desempenho**. 117-120, 2018.

PAPAPANAGIOTOU, L., CHEILLA, V. NDBench: Benchmarking microservices at scale. **ArXiv preprint arXiv**. 2018.

CATTELL, R. Scalable SQL and NoSQL data stores. **ACM Sigmod Record**. 12-27, 2011.