Avaliação de Performance para Fornecer StaaS a Dispositivos IoT em Ambiente Fog Computing

José dos Santos Machado, Danilo Souza Silva, Raphael Silva Fontes, Adauto Cavalcante Menezes, Edward David Moreno, Admilson de Ribamar Lima Ribeiro

Departamento de Computação, Universidade Federal de Sergipe – UFS – Brasil

jsmac18@hotmail.com, {danilosilva.se, raphaelf.ti, adauto.cavalcant, edwdavid}@gmail.com, admilson@ufs.br

Resumo. *Este trabalho apresenta a Fog Computing, sua contextualização teórica, os trabalhos correlatos e realiza avaliação de uma Fog Computing, para fornecer StaaS (Storage as a Service), a dispositivos IoT utilizando plataformas de sistemas embarcados e compara seus resultados com os obtidos por um servidor de alto desempenho. Os resultados demonstraram que a implementação desse serviço em dispositivos de sistemas embarcados pode ser uma boa alternativa para reduzir um desses problemas, no caso, o armazenamento de dados, que atinge hoje os dispositivos IoT.*

1. Introdução

Nas últimas décadas, observa-se que serviços de computação como armazenamento, processamento de dados e controle foram transferidos para a "nuvem". A oportunidade de computação ilimitada na nuvem permite que os usuários finais acessem amplas informações facilmente, também é possível visualizar que os dispositivos móveis e sensores, como *smartphones*, se tornaram poderosos equipamentos, o que levou ao surgimento de novos sistemas e aplicações.

Estes sistemas e aplicações introduzem novas demandas funcionais em computação e redes que a "nuvem" sozinha não pode atender. Neste cenário percebe-se que a computação local na borda da rede é muitas vezes necessária [7]. Por exemplo, para processar dados em tempo real, criar contextos de reconhecimento de localização a partir de sensores locais e maximizar a eficiência das comunicações sem fio na borda da rede. Em geral, a nuvem está muito longe dos dispositivos para satisfazer requisitos de latência e, é muito centralizada para lidar com a heterogeneidade e diversidade contextual em uma área local [14].

Para ultrapassar estas limitações, porções da capacidade de computação da nuvem podem ser deslocados para a borda da rede e formam um ambiente de computação local, isto é, uma "*Fog Computing*" chamada também de "nevoeiro" [7]. Ao distribuir os serviços de computação e de rede mais próximos de onde os dados do usuário são gerados, a *Fog* atende melhor às demandas emergentes [13].

A *Fog Computing* apresenta uma nova arquitetura que "leva processamento para os dados", enquanto a nuvem "leva os dados para o processamento" [1]. Dessa maneira dispositivos de borda e dispositivos móveis podem estar interligados dentro de uma rede local e executar colaborativamente tarefas de armazenamento, processamento de dados de rede e de controle [4].

A *Fog Computing* pode vir a resolver muitos problemas da Internet das Coisas (IoT), por exemplo, os serviços da *Fog* serão capazes de melhorar a largura de banda e as restrições de custo das comunicações de longo alcance. No entanto, muitos desafios ainda permanecem na computação em *Fog*, como modelar uma arquitetura de sistema para a *Fog*; como implementar, organizar e gerenciar dispositivos da *Fog*; como a *Fog* interage com a nuvem e com os dispositivos; e como gerenciar a conectividade física e lógica da *Fog*; entre outros.

Este trabalho apresenta o conceito da *Fog Computing*, os trabalhos correlatos e realiza a análise do fornecimento StaaS (*Storage as a Service*), a dispositivos IoT utilizando plataformas de sistemas embarcados em um ambiente *Fog Computing* e compara seus resultados com os obtidos por um servidor de alto desempenho.

O artigo está organizado em seis seções, a seção 2 apresenta o conceito e características da *Fog Computing*, na seção 3 é dedicada a revisão da literatura, a seção 4 temos o cenário de teste, na seção 5 avaliação e resultados e por fim na seção 6 as conclusões e trabalhos futuros.

2. Fog Computing

Devido à latência de rede frequentemente imprevisível, especialmente em um ambiente móvel, muitas vezes a computação em nuvem não pode atender aos requisitos rigorosos de latência, segurança e privacidade dos aplicativos em área restrita geograficamente [16]. Por outro lado, a crescente quantidade de dados gerados por dispositivos e sistemas, com poucos recursos pode se tornar impraticável para transportar dados através de redes para nuvens remotas [11].

Para isso, surgiu um novo paradigma, a *Fog Computing*. O conceito de computação em *Fog* foi adotado pela *Cisco Systems* como um novo paradigma em 2012 [4]. A *Fog Computing* é a computação em nuvem que distribuirá serviços avançados de computação, armazenamento, rede e gerenciamento mais próximos dos usuários finais, enviando informações dos dispositivos IoT para *Cloud Computing*, formando assim uma plataforma distribuída e virtualizada, assim, também é referido como computação de borda [6].

2.1 Características da Fog Computing

A computação *Fog* é um paradigma inovador que realiza computação distribuída, serviços de rede e armazenamento, além da comunicação entre *Cloud Computing Data Centers* até os dispositivos ao longo da borda da rede. Essa comunicação amplia as operações e serviços inerentes à computação em nuvem, permitindo assim uma nova geração de aplicativos [11]. A principal função é filtrar e agregar dados para os centros de dados da *Cloud* e aplicar inteligência lógica a dispositivos finais [7]. A figura 1 mostra a localização entre *Fog Computing* e *Cloud Computing*. A figura 2 apresenta arquitetura da *Fog Computing* com a sua localização.

Fig. 1 - Fog Localizada Entre Borda e Nuvem [14]

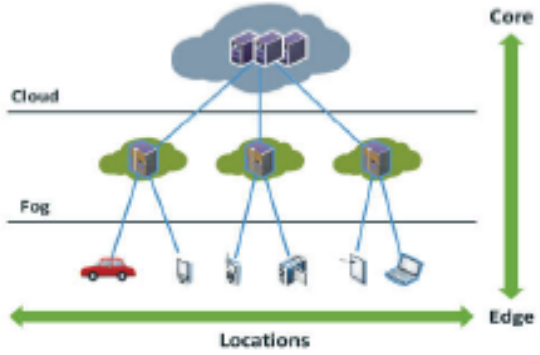
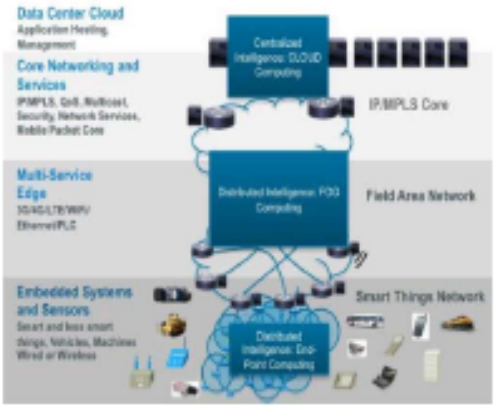


Fig. 2 - Arquitetura da Fog Computing [16]



Fog Computing é semelhante à computação em nuvem em muitos aspectos, no entanto, pode ser diferenciado do anterior pelo fato de estar próximo dos dispositivos finais, a distribuição espacial geograficamente menor e a possibilidade de apoio à mobilidade [7]. Como o processamento baseado em *Fog* ocorre ao longo da borda da rede, os resultados finais refletem uma percepção de localização altamente melhorada, baixa latência e Qualidade de Serviço (QoS), em aplicações de streaming e tempo real, os nós de nevoeiro são dispositivos heterogêneos, que vão desde servidores, pontos de acesso, roteadores de borda até dispositivos finais como telefones celulares, relógios inteligentes, sensores e etc [1].

2.2 Armazenamento como Serviço (StaaS)

A computação em nuvem e, em particular, os serviços de armazenamento em nuvem tornaram-se uma parte cada vez mais importante do setor de tecnologia da informação nos últimos tempos. O número de opções de armazenamento em nuvem está aumentando, com a maioria dos fornecedores fornecendo uma quantidade variada de armazenamento livre antes de cobrar por níveis de armazenamento maiores, devido ao número crescente desses serviços disponíveis, muitos pesquisadores usaram a frase *Storage as a Service* (StaaS), como uma extensão do *Software as a Service*, para descrever esse tipo de serviço [9].

3. Trabalhos Analisados

Um total de oito artigos foram identificados na implementação da *Fog Computing*, para melhorar alguns dos serviços da integração entre a *Cloud* e os dispositivos IoT, apresentaremos seus resultados e seus respectivos trabalhos futuros de pesquisas.

ZHU *et al.* (2013) [16], apresentaram a otimização da web dentro do contexto *Fog Computing*. Aplicaram métodos existentes para otimização da web de uma maneira inovadora, de tal forma que, esses métodos podem ser combinados com conhecimento exclusivo que está disponível apenas nos nós de borda (*Fog*). Como trabalho futuro sinalizaram aplicar seus conceitos propostos para desenvolver um sistema de prova de conceito.

No trabalho de CRACIUNESCU *et al.* (2015) [2], apresentaram uma implementação em laboratório de e-Saúde, onde o processamento em tempo real é realizado pelo PC doméstico, enquanto os metadados extraídos são enviados para a nuvem para processamento posterior. Como trabalho futuro sinalizaram adicionar mais sensores e mais dispositivos *off-the-shelf*, que são atualmente *mainstream*, e os dados de fusão desses dispositivos.

Em [15] VERBA *et al.* (2016), analisaram as plataformas existentes e suas deficiências, bem como propuseram uma plataforma de gateway modular, baseada em mensagens que permite o agrupamento de *gateways* e a abstração dos detalhes do protocolo de comunicação periférica. E sinalizaram como trabalho futuro desenvolver um ambiente de laboratório inteligente com diversos dispositivos mais complexos e cenários de controle para testar completamente o sistema.

FAN *et al.* (2016) [3], apresentaram a capacidade do recurso de *Web Caching* adicionado ao dispositivo de borda para servir como servidor *proxy* de armazenamento em cache, para obter mais armazenamento em cache. Os dispositivos finais também são explorados para fornecer algum espaço de cache. Como trabalho futuro sinalizaram adicionar mais funcionalidades ao dispositivo de borda, como a segurança e implementar o trabalho no mundo real.

No trabalho de HAO *et al.* (2017) [5], apresentaram um design do WM-FOG, uma estrutura de computação para ambientes *Fog*, que engloba essa arquitetura de *software* e avalia seu protótipo do sistema. Como trabalho futuro sinalizaram adicionar mais recursos ao WM-FOG para melhor atender às aplicações de computação em *Fog*.

Em [8] LI *et al.* (2017), discutiram dois conceitos de codificações recentemente propostos, códigos de mínima largura de banda e códigos de mínima latência, e ilustram seus impactos na computação em *Fog*, também analisaram uma estrutura de codificação unificada que inclui as duas técnicas de codificação acima descritas. Como trabalho futuro sinalizaram a necessidade de pesquisar computação heterogênea; redes com topologia de camada múltipla e estruturada; tarefas de computação em várias etapas; custos de computação codificados; verificar a computação distribuída; explorar as estruturas algébricas das tarefas computacionais; aplicações pesadas de comunicação e nós de *Fog plug-and-play*.

OSANAIYE *et al.* (2017) [11], apresentaram uma abordagem conceitual de migração em tempo real de pré cópia para a migração de VM e sinalizou como trabalho futuro a implantação do *framework* no mundo real ou ambiente de teste, com o objetivo de sua validação.

E por fim, POPENTIU-VLADICESCU et al. (2017) [12], analisaram os modelos de arquiteturas e práticas existentes em *Fog Computing* visando a confiabilidade e segurança dos sistemas de *Fog*, uma abordagem considerada integradora de três componentes da confiabilidade do sistema: a confiabilidade dos nós, a confiabilidade da rede e a confiabilidade do *software*, a arquitetura de referência *OpenFog* e os esquemas AJIA e BDSC. Como trabalho futuro sinalizaram resolver problemas técnicos e de algoritmos no paradigma de *Fog Computing*.

No quadro 1 os dados dos artigos analisados são sumarizados e comparados com este trabalho, foram organizados na ordem crescente cronologicamente para demostrar a

evolução em relação ao tema *Fog Computing*. Os artigos foram comparados quanto a implementação da *Fog Computing*, a utilização de plataforma de *Cloud*, utilização de dispositivo embarcado, uso de técnicas ou métodos de avaliação (*Benchmark*) e a implementação do StaaS.

Tabela 1 - Comparação Entre os Trabalhos

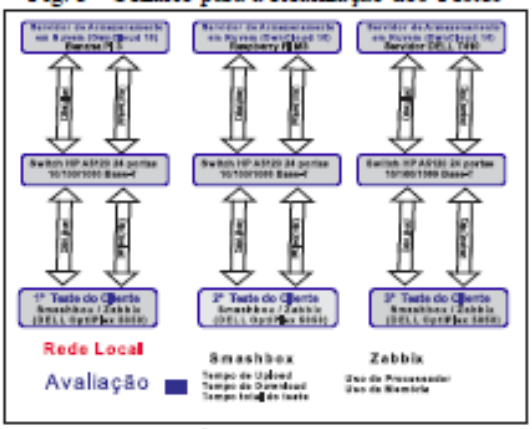
Artigo	Fog Computing	Plataforma Cloud	Dispositivo Embarcado	Benchmark	StaaS
[16] ZHU (2013)	✔				
[2] CRACIUNESCU (2015)	✔	✔	✔		
[15] VERBA (2016)	✔	✔	✔	✔	
[3] FAN (2016)	✔			✔	
[5] HAO (2017)	✔	✔		✔	
[8] LI (2017)	✔			✔	
[11] OSANAIYE (2017)	✔				
[12] POPENTIU-VLADICESCU (2017)	✔				
ESTE TRABALHO	✔	✔	✔	✔	✔

Fonte: Própria do Autor (2018)

4. Coleta de Dados

Para realizar a prototipação deve-se pressupor a existência de um modelo computacional idêntico ao ambiente real de produção. Na literatura, uma boa descrição para análise de desempenho em sistemas de nuvem para fornecer serviço de armazenamento StaaS (*Storage as a Service*) foi encontrada, como exemplo é possível citar o trabalho de MRÓWCZYŃSKI *et al.* (2017) [10]. A figura 3 ilustra a arquitetura do cenário para a realização dos testes de avaliação.

Fig. 3 - Cenário para a Realização dos Testes



Fonte: Própria do Autor (2018)

Tabela 2 - Abreviaturas dos Testes

Nome	Abreviação	Quantidade de arquivos	Tamanho arquivo	Volume total
Test0	0/1/1	1	1B	1B
Test1	0/1/100000000	1	100Mb	100Mb
Test2	0/10/10000000	10	10Mb	100Mb
Test3	0/1000/10000	1000	10Kb	10Mb
Test4	0/1/500000000	1	500Mb	500Mb

Fonte: Própria do Autor (2018)

Foram realizados cinco tipos de diferentes testes para avaliar o desempenho dos equipamentos analisados, para o fornecimento do serviço StaaS de forma sequencialmente. A tabela 2 informa abreviaturas dos testes (onde-se, 1° número equivale a quantidade de diretório, 2° quantidade de arquivo e o 3° volume do arquivo) e sua correspondente distribuição de arquivos.

4.1 Benchmark Smashbox

O Smashbox é uma estrutura de *benchmarking* e monitoramento para sincronização de arquivos e serviços de compartilhamento, permitindo aos provedores de serviços monitorar o status operacional de seus serviços, entendendo o comportamento do serviço sob diferentes tipos de carga e com diferentes locais de rede para a sincronização de clientes [10]. O *container* do sistema Smashbox está disponibilizado na condição pública no Docker Hub, no endereço https://hub.docker. com/r/ jsmac/smashbox/.

5. Avaliação e Resultados

Essa etapa consiste em: realizar a montagem do cenário de teste com os dispositivos de sistemas embarcados e o servidor de forma individual; efetuar a abordagem dos *softwares* necessários nos dispositivos para a elaboração do experimento; proceder com o processo de coleta dos dados do tempo de *upload*, tempo de *download* e tempo total de cada teste realizado com o *benchmark* Smashbox, as métricas de utilização da CPU e memória, equivalem ao máximo obtidos nas 30 repetições dos testes. A figura 4 ilustra o cenário real em que os testes foram realizados.

Fig. 4 - Cenário Real de Teste

Fonte: Própria do Autor (2018)

Tabela 3 - Diferentes Implementações

Equipamentos utilizados na avaliação				
EQUIP.	S.O	VIRTUAL	RAM	REDE
Banana PI M3	Ubuntu Server 16.04	Não suporta	2 Gb	1000 Mb/s
Raspberry PI 3	Raspbian Stretch Lite 9	S/ virtualização	1 Gb	100 Mb/s
Raspberry PI 3	Raspbian Stretch Lite 9	Docker	1 Gb	100 Mb/s
Raspberry PI 3	Ubuntu MATE 16.04	Docker	1 Gb	100 Mb/s
Servidor Dell T410	Ubuntu Server 16.04	VMware ESXi + Docker	16 Gb	1000 Mb/s
Desktop Dell OptiPlex 5050	Windows 10	Docker	16 Gb	1000 Mb/s

Fonte: Própria do Autor (2018)

Foram implementados 5 (cinco) cenários com diferentes sistemas operacionais, com e sem a implementação da virtualização Docker *Container*, isso possibilitou analisar o melhor conjunto implementado e o impacto da virtualização em dispositivos de sistemas embarcados. A tabela 3 mostra as diferentes implementações utilizadas com diferentes sistemas operacionais.

Totalizaram-se 750 coletas de dados, 30 repetições para os 5 diferentes testes, utilizando as 5 diferentes implementações entre os equipamentos testados e diferentes sistemas operacionais. Só foram aceitos para análise os testes concluídos sem erros. O tempo encontra-se em segundos. A tabela 4 mostra os resultados obtidos por todas as implementações para o Test0 na transferência de 1 arquivo do tamanho de 1 byte.

5.1 Test0 0/1/1

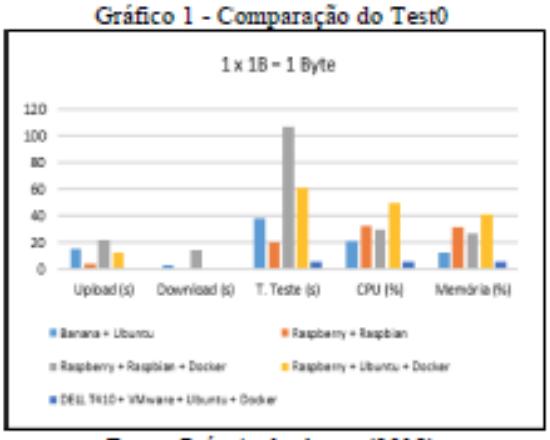
Nota-se que nesse teste todas as implementações obtiveram resultados melhores no tempo de *download* em relação ao tempo de *upload*. Também se observa que as implementações nos dispositivos de sistemas embarcados obtiveram um valor muito elevado na métrica de desvio padrão, o valor. O gráfico 1 mostra melhor visualmente a comparação entre todas as implementações.

Tabela 4 – Resultado do Test0

Tempo segundos	SMASHBOX				ZABBIX	
	MÉDIA	DESVIO	MIN	MAX	CPU %	MEM %
BANANA PI M3 + UBUNTU SERVER						
T. UPL	15,13	5,54	3,00	26,00		
T. DWL	3,33	2,01	2,00	9,00	21,50	12,50
TOTAL	38,23	8,65	21,00	57,00		
RASPBERRY PI 3 + RASPBIAN STRETCH LITE						
T. UPL	4,20	5,26	1,00	17,00		
T. DWL	1,73	2,30	0,82	13,00	32,42	31,39
TOTAL	20,40	22,23	6,00	62,00		
RASPBERRY PI 3 + RASPBIAN LITE + DOCKER						
T. UPL	22,60	3,76	14,00	31,00		
T. DWL	14,53	2,61	7,00	20,00	29,36	26,55
TOTAL	106,60	10,41	73,00	122,00		
RASPBERRY PI 3 + UBUNTU MATE + DOCKER						
T. UPL	12,83	1,64	9,00	17,00		
T. DWL	7,03	1,27	4,00	11,00	49,93	41,09
TOTAL	60,87	2,15	56,00	64,00		
DELL T410 + VMWARE + UBUNTU + DOCKER						
T. UPL	1,00	0,00	1,00	1,00		
T. DWL	0,99	0,02	0,90	1,00	6,40	6,00
TOTAL	6,00	0,26	5,00	7,00		

Fonte: Própria do Autor (2018)

Gráfico 1 - Comparação do Test0



Fonte: Própria do Autor (2018)

A implementação do servidor DELL T410 é considerada a ideal, afinal é um equipamento de alto desempenho, porém com um alto custo aquisitivo, percebe-se que os seus resultados foram melhores em quase todos os aspectos analisados em relação às outras implementações.

5.2 Test1 0/1/100000000

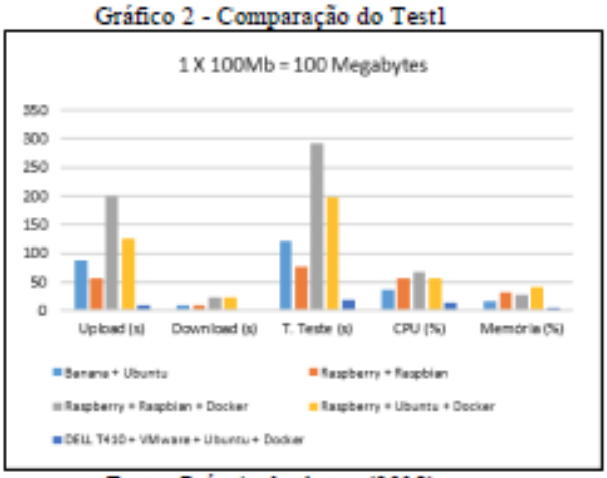
O Test1 consiste em avaliar a transferência do volume de um arquivo com o tamanho de 100 megabytes. A tabela 5 mostra os resultados obtidos por todas as implementações para o Test1 na transferência do arquivo.

Tabela 5 – Resultado do Test1

Tempo segundos	SMASHBOX				ZABBIX	
	MÉDIA	DESVIO	MIN	MAX	CPU %	MEM %
BANANA PI M3 + UBUNTU SERVER						
T. UPL	88,50	14,45	67,00	125,00		
T. DWL	10,60	4,58	6,00	27,00	36,31	15,50
TOTAL	121,63	19,24	95,00	159,00		
RASPBERRY PI 3 + RASPBIAN STRETCH LITE						
T. UPL	56,93	26,51	30,00	144,00		
T. DWL	9,70	1,78	9,00	18,00	56,21	33,06
TOTAL	77,67	34,84	44,00	191,00		
RASPBERRY PI 3 + RASPBIAN LITE + DOCKER						
T. UPL	199,20	16,61	148,00	223,00		
T. DWL	23,93	5,56	14,00	42,00	67,60	28,76
TOTAL	290,90	25,80	174,00	317,00		
RASPBERRY PI 3 + UBUNTU MATE + DOCKER						
T. UPL	125,13	7,29	117,00	153,00		
T. DWL	23,57	4,28	17,00	32,00	55,96	40,43
TOTAL	197,13	9,67	184,00	237,00		
DELL T410 + VMWARE + UBUNTU + DOCKER						
T. UPL	9,23	0,43	9,00	10,00		
T. DWL	4,00	0	4,00	4,00	14,03	6,00
TOTAL	18,87	0,35	18,00	19,00		

Fonte: Própria do Autor (2018)

Gráfico 2 - Comparação do Test1



Fonte: Própria do Autor (2018)

A implementação Banana + Ubuntu obteve uma métrica melhor no quesito tempo médio de *upload* comparado a implementação do Raspberry com ubuntu e utilizando virtualização Docker, fato que não tinha ocorrido no teste Test0. Porém, seus resultados ficaram abaixo comparado com a implementação do Raspberry com raspbian sem virtualização. O gráfico 2 mostra melhor visualmente a comparação.

5.3 Test2 0/10/10000000

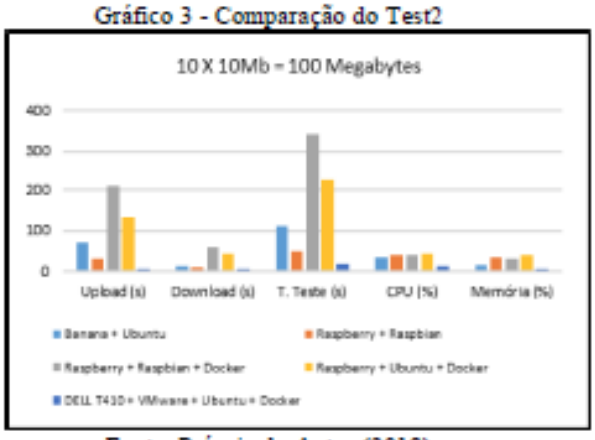
O test2 consiste em avaliar a transferência do volume de dez arquivos com o tamanho de 10 megabytes. A tabela 6 mostra os resultados obtidos por todas as implementações para o Test2 na transferência dos arquivos.

Tabela 6 – Resultado do Test2

Tempo segundos	SMASHBOX				ZABBIX	
	MÉDIA	DESVIO	MIN	MAX	CPU %	MEM %
BANANA PI M3 + UBUNTU SERVER						
T. UPL	70,83	13,28	49,00	111,00		
T. DWL	11,77	7,54	5,00	32,00	34,91	15,50
TOTAL	111,17	18,20	85,00	147,00		
RASPBERRY PI 3 + RASPBIAN STRETCH LITE						
T. UPL	30,63	29,74	16,00	158,00		
T. DWL	10,20	3,92	9,00	30,00	40,88	33,85
TOTAL	49,83	34,27	33,00	199,00		
RASPBERRY PI 3 + RASPBIAN LITE + DOCKER						
T. UPL	209,63	14,72	184,00	236,00		
T. DWL	59,37	8,273	50,00	95,00	41,13	30,37
TOTAL	341,77	19,49	307,00	414,00		
RASPBERRY PI 3 + UBUNTU MATE + DOCKER						
T. UPL	133,37	21,92	117,00	242,00		
T. DWL	42,40	7,96	34,00	80,00	41,96	39,50
TOTAL	225,70	21,93	212,00	330,00		
DELL T410 + VMWARE + UBUNTU + DOCKER						
T. UPL	7,40	0,89	6,00	9,00		
T. DWL	4,80	0,61	4,00	6,00	13,20	6,00
TOTAL	18,73	1,08	17,00	21,00		

Fonte: Própria do Autor (2018)

Gráfico 3 - Comparação do Test2



Fonte: Própria do Autor (2018)

Na implementação Raspberry + Raspbian + Docker, nota-se que continuou obtendo os piores resultados no teste, com a média 209,63 segundos de *upload* e 59,37 segundos de *download*, e uma média de tempo de conclusão dos testes muito elevada 341,77 segundos. Percebe-se também que a implementação chegou a utilizar um nível razoável de processamento no teste 41,13% e com o uso de memória mediano 30,37%. O gráfico 3 mostra melhor visualmente a comparação.

5.4 Test3 0/1000/10000

O test3 consiste em avaliar a transferência do volume de mil arquivos com o tamanho de 10 kilobytes, acredita-se que esse teste seja a simulação mais próxima da realidade do funcionamento de uma *Fog Computing*, pequenos arquivos, porém em grande quantidade. A tabela 7 mostra os resultados obtidos por todas as implementações.

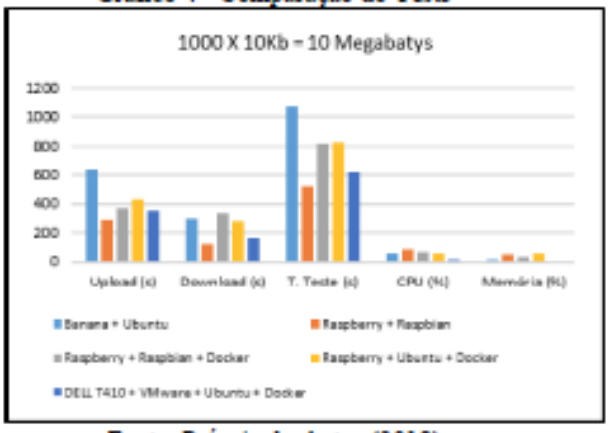
A implementação Raspberry + Raspbian surpreendeu, superando até mesmo a implementação do servidor DELL T410. Todavia nota-se que o desvio padrão dos seus resultados apresentaram uma alta discrepância, isso o torna um serviço pouco preciso. O gráfico 4 mostra melhor visualmente a comparação.

Tabela 7 – Resultado do Test3

Tempo segundos	SMASHBOX				ZABBIX	
	MÉDIA	DESVIO	MIN	MAX	CPU %	MEM %
BANANA PI M3 + UBUNTU SERVER						
T. UPL	640,67	459,94	418,00	2496,00	59,67	18,00
T. DWL	301,67	121,19	232,00	687,00		
TOTAL	1079,03	540,16	804,00	2889,00		
RASPBERRY PI 3 + RASPBIAN STRETCH LITE						
T. UPL	289,40	174,19	148,00	746,00	87,21	46,56
T. DWL	120,77	98,14	87,00	546,00		
TOTAL	521,50	212,90	346,00	1175,00		
RASPBERRY PI 3 + RASPBIAN LITE + DOCKER						
T. UPL	369,03	46,41	346,00	610,00	67,65	31,14
T. DWL	334,43	25,82	327,00	469,00		
TOTAL	812,60	56,57	779,00	1080,00		
RASPBERRY PI 3 + UBUNTU MATE + DOCKER						
T. UPL	430,57	81,83	335,00	685,00	61,65	60,82
T. DWL	278,13	210,58	106,00	1056,00		
TOTAL	821,17	292,95	553,00	1858,00		
DELL T410 + VMWARE + UBUNTU + DOCKER						
T. UPL	350,63	6,14	340,00	364,00	12,00	7,08
T. DWL	163,93	0,25	163,00	164,00		
TOTAL	620,63	6,36	610,00	635,00		

Fonte: Própria do Autor (2018)

Gráfico 4 - Comparação do Test3



Fonte: Própria do Autor (2018)

5.5 Test4 0/1/500000000

O Test4 consiste em avaliar a transferência do volume de um arquivo com o tamanho de 500 megabytes. A tabela 8 mostra os resultados obtidos por todas as implementações.

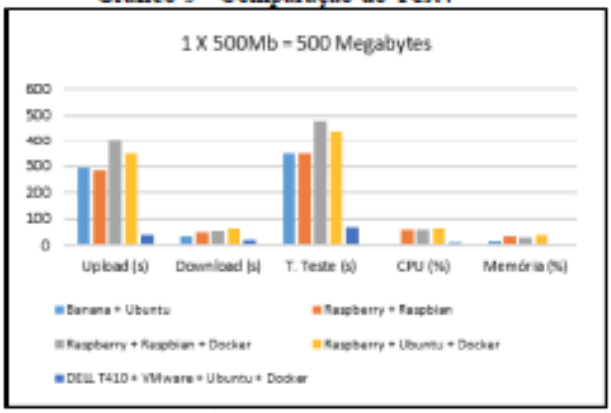
Na implementação Raspberry + Ubuntu + Docker obteve bons resultados quando comparado com a implementação de concorrência direta, Raspberry + Raspbian + Docker. Todavia nota-se que foi a implementação que obteve o mais alto nível do uso da memória do *hardware* 65,00%. O gráfico 5 mostra melhor visualmente a comparação.

Tabela 8 – Resultado do Test4

Tempo segundos	SMASHBOX				ZABBIX	
	MÉDIA	DESVIO	MIN	MAX	CPU %	MEM %
BANANA PI M3 + UBUNTU SERVER						
T. UPL	294,00	24,62	237,00	340,00	40,92	17,00
T. DWL	36,33	12,69	21,00	81,00		
TOTAL	347,47	31,03	271,00	394,00		
RASPBERRY PI 3 + RASPBIAN STRETCH LITE						
T. UPL	281,83	84,91	174,00	449,00	57,05	33,96
T. DWL	49,40	5,93	44,00	69,00		
TOTAL	345,43	94,24	226,00	544,00		
RASPBERRY PI 3 + RASPBIAN LITE + DOCKER						
T. UPL	394,37	146,64	191,00	886,00	59,41	30,30
T. DWL	55,30	8,26	47,00	81,00		
TOTAL	476,80	158,12	254,00	1018,00		
RASPBERRY PI 3 + UBUNTU MATE + DOCKER						
T. UPL	346,30	168,15	190,00	850,00	65,00	38,95
T. DWL	63,97	25,32	48,00	161,00		
TOTAL	437,03	188,62	258,00	961,00		
DELL T410 + VMWARE + UBUNTU + DOCKER						
T. UPL	40,30	3,48	37,00	51,00	12,00	7,08
T. DWL	19,23	3,08	16,00	27,00		
TOTAL	67,47	5,96	61,00	80,00		

Fonte: Própria do Autor (2018)

Gráfico 5 - Comparação do Test4



Fonte: Própria do Autor (2018)

Diferentes aplicações de computação em *Fog* foram sugeridas na literatura. Segundo OSANAIYE *et al.* (2017) [11], as categorias das aplicações de computação em *Fog* são divididas em 3: (i) Aplicações em tempo real; (ii) Aplicações quase em tempo real; (iii) Aplicações introduzidas em redes.

Por isso, definir o nível aceitável do fornecimento do serviço de armazenamento StaaS, em uma emergente tecnologia em que se trata o ambiente *Fog Computing* é uma tarefa complexa e subjetiva, dependente da classificação da aplicação.

6. Conclusão e Trabalhos Futuros

Este trabalho apresenta o conceito da *Fog Computing*, sua contextualização teórica, os trabalhos correlatos, realiza análise de uma *Fog Computing*, para fornecer StaaS (*Storage as a Service*), a dispositivos IoT utilizando plataformas de sistemas embarcados e compara seus resultados com os obtidos por um servidor de alto desempenho. Foram realizados cinco (5) implementações de diferentes combinações de *softwares* e *hardwares*, e analisa seus resultados com a finalidade de encontrar a melhor opção para disponibilizar o serviço de armazenamento StaaS em um ambiente *Fog Computing*.

Os resultados satisfizeram as expectativas, na avaliação do teste Test3 0/1000/10000 (transferência de 1000 arquivos de 10 *Kilobytes*) a implementação Raspberry + Raspbian surpreendeu, obtendo ótimos resultados, superando até mesmo a implementação do servidor DELL T410. Nota-se que este tipo de implementação pode satisfazer as aplicações classificadas em aplicações quase em tempo real e introduzida na rede, não sendo adequada para as aplicações classificadas como aplicações em tempo real, devido as implementações nos dispositivos de sistemas embarcados obterem valores muito elevados nas métricas de desvio padrão, tornando o serviço pouco preciso.

Portanto percebe-se que a implementação desse serviço em dispositivos de sistemas embarcados pode ser uma boa alternativa para reduzir um desses problemas, no caso, o armazenamento de dados, que atinge hoje os dispositivos IoT, servindo como *Fog Computing* e sendo implantado num dispositivo de plataforma embarcada de baixo custo, ao invés de usar potentes e caros servidores para exercer essa função.

6.1 Trabalhos Futuros

Com a finalidade em dar continuidade a essa pesquisa, acredita-se que esse trabalho abriu vários cenários para futuros trabalhos, sendo:

- Analisar o serviço StaaS, com diferentes dispositivos embarcados e diferentes plataformas de serviço de armazenamento em nuvem, não utilizados neste trabalho;
- Realizar a mesma avaliação utilizando *cluster* com dispositivos de sistemas embarcados, usando a virtualização Dorcker.

7. Referências

[1] AL-DOGHMAN, Firas et al. A review on Fog Computing technology. In: Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on. IEEE, 2016. p. 001525-001530.

[2] CRACIUNESCU, Razvan et al. Implementation of Fog computing for reliable E-health applications. In: Signals, Systems and Computers, 2015 49th Asilomar Conference on. IEEE, 2015. p. 459-463.

[3] FAN, Chih-Tien et al. Web Resource Cacheable Edge Device in Fog Computing. In: Parallel and Distributed Computing (ISPDC), 2016 15th International Symposium on. IEEE, 2016. p. 432-439.

[4] HAJIBABA, Majid; GORGIN, Saeid. A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing. CIT. Journal of Computing and Information Technology, v. 22, n. 2, p. 69-84, 2014.

[5] HAO, Zijiang et al. Challenges and Software Architecture for Fog Computing. IEEE Internet Computing, v. 21, n. 2, p. 44-53, 2017.

[6] JAIN, Akshay; SINGHAL, Priyank. Fog computing: Driving force behind the emergence of edge computing. In: System Modeling & Advancement in Research Trends (SMART), International Conference. IEEE, 2016. p. 294-297.

[7] MACHADO, José dos Santos; MORENO, Edward David; RIBEIRO, Admilson de Ribamar Lima. A Review of Computing Fog and its Research Challenges. Journal on Advances in Theoretical and Applied Informatics, [S.l.], v. 3, n. 2, p. 32-39, dec. 2017. ISSN 2447-5033.

[8] LI, Songze; MADDAH-ALI, Mohammad Ali; AVESTIMEHR, A. Salman. Coding for Distributed Fog Computing. IEEE Communications Magazine, v. 55, n. 4, p. 34-40, 2017.

[9] MARTINI, Ben; CHOO, Kim-Kwang Raymond. Cloud storage forensics: ownCloud as a case study. Digital Investigation, v. 10, n. 4, p. 287-299, 2013.

[10] MRÓWCZYŃSKI, Piotr et al. Benchmarking and monitoring framework for interconnected file synchronization and sharing services. Future Generation Computer Systems, 2017.

[11] OSANAIYE, Opeyemi et al. From cloud to fog computing: A review and a conceptual live VM migration framework. IEEE Access, 2017.

[12] POPENTIU-VLADICESCU, Florin; ALBEANU, Grigore. Software reliability in the fog computing. In: Innovations in Electrical Engineering and Computational Technologies (ICIEECT), 2017 International Conference on. IEEE, 2017. p. 1-4.

[13] MCMILLIN, Bruce; ZHANG, Tao. Fog Computing for Smart Living. Computer, v. 50, n. 2, p. 5-5, 2017.

[14] STOJMENOVIC, Ivan et al. An overview of Fog computing and its security issues. Concurrency and Computation: Practice and Experience, 2015.

[15] VERBA, Nandor et al. Platform as a service gateway for the Fog of Things. Advanced Engineering Informatics, 2016.

[16] ZHU, Jiang et al. Improving web sites performance using edge servers in fog computing architecture. In: Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on. IEEE, 2013. p. 320-323.

[17] ALRAWAIS, Arwa et al. Fog Computing for the Internet of Things: Security and Privacy Issues. IEEE Internet Computing, v. 21, n. 2, p. 34-42, 2017.

[18] BABU, Shaik Masthan; LAKSHMI, A. Jaya; RAO, B. Thirumala. A study on cloud based internet of things: Cloudiot. In: Communication Technologies (GCCT), 2015 Global Conference on. IEEE, 2015. p. 60-65.

[19] BOTTA, Alessio et al. Integration of cloud computing and internet of things: a survey. Future Generation Computer Systems, v. 56, p. 684-700, 2016.

[20] CHEN, Songqing; ZHANG, Tao; SHI, Weisong. Fog Computing. IEEE Internet Computing, v. 21, n. 2, p. 4-6, 2017.

[21] CHIANG, Mung et al. Clarifying Fog Computing and Networking: 10 Questions and Answers. IEEE Communications Magazine, v. 55, n. 4, p. 18-20, 2017.

[22] DÍAZ, Manuel; MARTÍN, Cristian; RUBIO, Bartolomé. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. Journal of Network and Computer Applications, v. 67, p. 99-117, 2016.

[23] KUMAR, Praveen; ZAIDI, Nabeel; CHOUDHURY, Tanupriya. Fog computing: Common security issues and proposed countermeasures. In: System Modeling & Advancement in Research Trends (SMART), International Conference. IEEE, 2016. p. 311-315.

[24] LINTHICUM, David S. Connecting Fog and Cloud Computing. IEEE Cloud Computing, v. 4, n. 2, p. 18-20, 2017.

[25] NWOBODO, Ikechukwu (2015). A Comparison of Cloud Computing Platforms. International Symposium on Circuits and Systems (ISCAS). Lisbon, Portugal, 24-27 May 2015. Atlantis Press.

[26] PRINCY, S. Emima; NIGEL, K. Gerard Joe. Implementation of cloud server for real time data storage using Raspberry Pi. In: Green Engineering and Technologies (IC-GET), 2015 Online International Conference on. IEEE, 2015. p. 1-4.

[27] LONGO, Francesco et al. Stack4things: An openstack-based framework for iot. In: Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on. IEEE, 2015. p. 204-211.

[28] ZHANG, Qi; CHENG, Lu; BOUTABA, Raouf. Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, v. 1, n. 1, p. 7-18, 2010.