ANALISA SUMBER DAYA MEMORI UNTUK IMPLEMENTASI IPFS (*INTERPLANETARY FILE SYSTEM)* PADA *SMART CONTRACT* ETHEREUM

MEMORY ANALYSIS FOR IPFS (INTERPLANETARY *FILE* SYSTEM) IMPLEMENTATION ON ETHEREUM *SMART CONTRACT*S

Tiara Sabrina[1], Avon Budiyono[2], Adityas Widjajarto[3]

[1,2,3]Prodi S1 Sistem Informasi, Fakultas Rekayasa Industri, Universitas Telkom
[1]**sabrinatiara@student.telkomuniversity.ac.id**, [2]**avonbudi@telkomuniveristy.co.id**,
[3]**adtwjrt@telkomuniversity.ac.id**

Abstrak

***Smart contract* adalah kesepakatan antara dua entitas yang dituangkan dalam kode program. Seluruh transaksi *Smart contract* disimpan di Blockchain. Blockchain adalah teknologi *peer-to-peer* terdistribusi untuk menyimpan dan mendistribusikan data *digital* seperti cryptocurrency dan smart contract dengan adanya kerahasiaan, integritas dan keaslian data. Akan tetapi Blockchain tidak cocok untuk menyimpan data dalam jumlah besar maka banyak *developer* saat ini membuat suatu DApp yang mengintegrasikan IPFS pada *Smart contract* Ethereum. File akan disimpan di IPFS sedangkan Blockchain hanya menyimpan *hash file* dari IPFS untuk dapat mengaksesnya kembali. Pada penelitian ini dilakukan pengukuran pemakaian memori dan CPU saat menjalankan DApp. Terdapat dua parameter yang mempengaruhi pemakaian memori dan CPU yaitu ukuran file dan jumlah node yang berinteraksi. Hasil pengujian akan digunakan sebagai tolak ukur dalam proses perencaanaan kapasitas memori dan CPU dalam menjalankan DApp yang mengintegrasikan IPFS pada Smart Contract. Perencanaan kapasitas hardware memori dan CPU diperlukan agar sistem yang dibangun bisa bekerja dengan baik sesuai kebutuhannya.**
Kata Kunci : ***IPFS, Blockchain, Ethereum, Smart contract, Memori, RAM***

Abstract

Smart contract is an agreement between two entities as outlined in the program code. All Smart contract transactions are stored on the Blockchain. Blockchain is a distributed peer-to-peer technology for storing and distributing digital data such as cryptocurrency and smart contracts with the confidentiality, integrity and authenticity of data. However, Blockchain is not suitable for storing large amounts of data, so many developers now make a DApp (Decentralized Application) that integrates IPFS on Smart contract Ethereum. Files will be stored on IPFS while the Blockchain only stores the hashes of files stored on IPFS to be able to access them again. In this study, memory usage measurements were carried out when running DApp through the file upload process. The test results prove that the increase in RAM usage in each file upload process is influenced by file size and the number of nodes that interact with the system. The memory usage test results will be used as a benchmark in the capacity planning process so that the DApp web system can work properly according to its needs.

Key Word : IPFS, Blockchain, Ethereum, Smart contract, Memori, RAM

1. Pendahuluan

Saat ini hampir seluruh kegiatan bisnis dilakukan melalui pembuatan kontrak tertulis. Kontrak adalah ikatan kesepakatan/perjanjian tertulis yang mendefinisikan hak dan tanggung jawab masing-masing pihak. Kontrak tertulis memiliki beberapa kelemahan yaitu kontrak dapat hilang dan rusak, salah satu pihak melakukan kecurangan, dan kontrak tertulis lebih menghabisan biaya dan waktu (Oscar, 2018). Blockchain menciptakan suatu terobosan yaitu Ethereum dengan fiturnya yaitu smart contract. Dengan smart contract pengembang dapat menulis kontrak digital melalui kode program. Transaksi kontrak pintar memiliki kredibilitas tinggi karena kontrak yang telah dibuat tidak dapat dilacak dan diubah. Smart Contract menjadi solusi permasalahan dari kontrak tertulis. (Atzei, Bartoletti dan Cimoli, 2017).

Smart contract berserta transaksinya disimpan pada Blockchain. (Dannen, 2017:12). Blockchain adalah teknologi *peer-to-peer* terdistribusi untuk menyimpan dan mendistribusikan data *digital* seperti *cryptocurrency*, *Smart contract*, properti, saham, berkas, atau apa pun yang berharga dengan adanya kerahasiaa, integritas dan keaslian data (Rajalakshmi, 2018:1437) Grech dan Camilleri (2017:22-23).

Akan tetapi Blockchain tidak cocok untuk menyimpan data dalam jumlah besar maka banyak *developer* saat ini membuat suatu *Web* DApp yang mengintegrasikan IPFS pada *Smart contract* Ethereum. File akan disimpan di IPFS sedangkan Blockchain hanya menyimpan *hash file* dari IPFS untuk dapat mengaksesnya kembali. (Rajalakshmi dkk, 2018). Dengan adanya integrasi IPFS pada *Smart contract*, biaya gas yang dikeluarkan untuk pembuatan contract dan pembuatan transaksi lebih murah dibandingkan dengan tanpa adanya integrasi IPFS pada *Smart contract* (Sinha dan Kaul, 2018:1209). Gas adalah sebagian kecil dari Ether yang digunakan untuk membayar para miners untuk melakukan proses mining (Ethos, 2018). Proses mining adalah proses menyimpan transaksi dalam suatu blok dan menambahkan blok ke *blockchain*. (Wang dkk, 2016:4-5)

Pada penelitian ini akan dibangun sebuah DApp berbasis *web* yang mengintegrasikan IPFS pada *Smart contract* Ethereum. Untuk menjalankan *Web* DApp yang telah dibuat pada sistem operasi membutuhkan sumber daya memori yaitu RAM dan CPU agar *web* DApp dapat beroperasi sebagai mana mestinya. Penelitian yang dilakukan untuk mengukur konsumsi RAM dan CPU yang digunakan *web* DApp. Beberapa parameter yang mempengaruhi pemakaian memori dan CPU yaitu ukuran file yang akan diunggah dan jumlah node yang berinteraksi dengan sistem. Jumlah pemakaian memori dan CPU yang dipakai oleh *web* DApp akan menjadi tolak ukur dalam mengalokasikan memori dan CPU yang sesuai dengan kebutuhan dari aktivitas user dan budget yang tersedia. Terlebih pasar menyajikan perangkat keras dengan spesifikasi yang beragam untuk memenuhi kebutuhan user. Semakin baik kualitas perangkat keras maka semakin mahal harga perangkat tersebut.

2. Dasar Teori

2.1 *Smart contract* Ethereum

Ethereum adalah *platform* berbasis Blockchain yang dibangun khusus untuk membuat dan menjalankan *Smart contract*. *Smart contract* merupakan protokol komputer yang dimaksudkan untuk memfasilitasi, memverifikasi, atau menegakkan negosiasi atau kinerja suatu kontrak secara *digital* melalui kode program YANG ditulis dalam bahasa EVM bytecode. *Smart contract* dan transaksi dibuat, disimpan dan dijalankan diatas Blockchain disetiap *node* jaringan untuk mempercepat proses komunikasi dan mencegah konflik kehilangan data (Dannen, 2017:12). Pengeksekusian *Smart contract* dipicu oleh sebuah transaksi dan setiap transaksi mengkonsumsi Ether. (Galal dan Youssef, 2018:2) Ada tiga alasan *user* melakukan transaksi di jaringan Ethereum, yaitu: (i) membuat kontrak baru; (ii) menjalankan fungsi kontrak; (iii) men-*transfer* eter ke kontrak atau ke pengguna lain (Atzei, Bartoletti dan Cimoli, 2017:1-3). Ethereum memiliki tiga jaringan untuk melakukan pengembangan dan pengujian bagi *developer* yaitu Ropsten, Kovan dan Rinkeby. (Kenneth, 2015)

Smart contract ditulis dalam bahasa EVM *bytecode*. Developer pada umumnya tidak menulis *Smart contract* menggunakan bahasa EVM bytecode melainkan bahasa pemograman Solidity (Parizi dkk, 2018). Solidity adalah bahasa pemrograman mirip Javascript yang digunakan untuk membuat *Smart contract*. Kode yang telah dibuat akan dikompilasi menjadi EVM byte-code agar dapat dieksekusi di jaringan Ethereum (Atzei, Bartoletti dan Cimoli, 2017:3).

2.3 Blockchain

Teknologi Blockchain berguna untuk menyimpan dan mendistribusikan data *digital* dengan adanya kerahasiaa, integritas dan keaslian data (Rajalakshmi, 2018:1437).Blockchain menggunakan struktur data *block-chain* untuk menyimpan dan memverifikasi data, menggunakan algoritma *distributed node concencus* untuk memperbarui data transaksi, menggunakan kriptografi asimetris untuk memastikan keamanan akses dan transimisi data (Chen, 2017:2652). Semua peserta (*node*) dalam jaringan Blockchain memiliki salinan *database* yang sama.

2.4 IPFS

IPFS adalah sebuah *distributed file system* pada jaringan *peer-to-peer* (P2P), berfungsi untuk menghubungkan semua perangkat komputasi dengan sistem *file* yang sama. IPFS mengkombinasikan ide-ide sukses dari sistem *peer-to-peer* sebelumnya yaitu DHT, BitTorrent, Git, dan SFS (Juan Benet, 2014:1-3). Setiap *node* diidentifikasi dengan kriptografi *hash* dari *public key* yang dibuat dengan kripto S / Kademlia (Oscar Wennergren dkk,2014:04). Jaringan IPFS menggunakan sistem *routing* yang berlandaskan DHT (*Distributed Hash Table*) yang dapat mengakomodasi jutaan *node* (Chen, 2017). IPFS mendistribusikan *file* melalui protokol bernama Bitswap. Pendistribusian *file* dilakukan dengan cara bertukar blok antara *peer*. Bitswap akan menyimpan history *file transfer* pada Bitswap *Ledger*. Semakin banyak *byte* yang tercatat dalam Bitswap *Legder* maka *node* tersebut semakin dipercaya oleh Bitswap(Juan Benet, 2014:04-06).

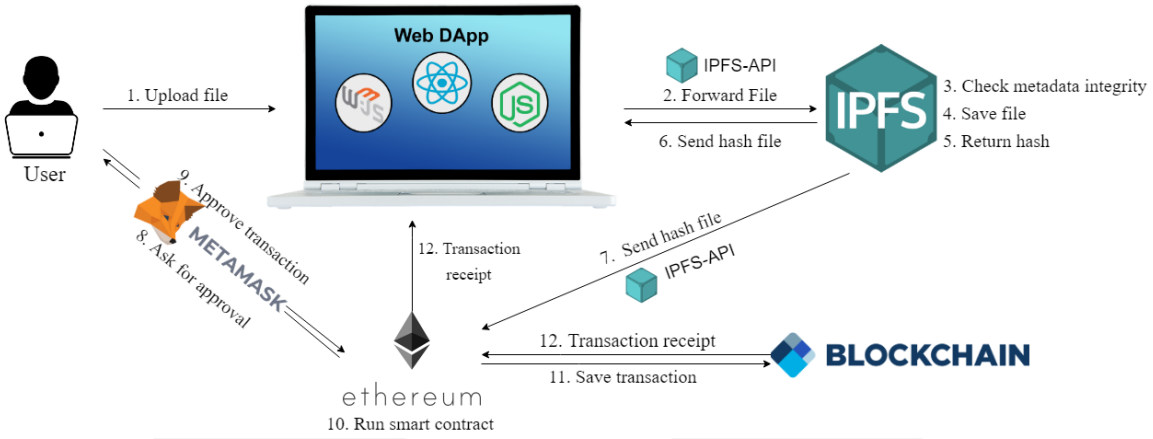
2.5 RAM (Read Only Memory)

RAM terbagi atas dua jenis yaitu *Static* RAM (SRAM) dan *Dynamic* RAM (DRAM). SRAM digunakan untuk menyimpan *cache* yang terdapat di dalam atau diluar chip CPU. DRAM digunakan untuk *main memory* (*physical* RAM) dan *frame buffer* dari *graphic system*. Biasanya SRAM hanya memiliki beberapa *megabyte* saja, sedangkan DRAM bisa mencapai ratusan bahkan ribuan *megabyte* (Bryant & O'Hallaron, 2013:9&561).

Jenis RAM yang menjadi fokus dalam penelitian ini adalah *main memory* (*physical* RAM). *Main memory* (*physical* RAM) merupakan perangkat penyimpanan sementara yang menampung program berserta data-datanya saat prosesor mengeksekusi program tersebut.

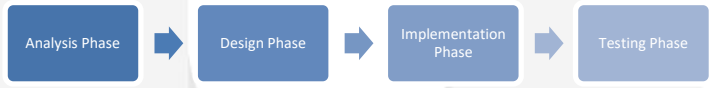
3. Arsitektur Sistem

Arsitektur sistem dibawah ini User membuka web DApp melalui browser dan mengunggah file melalui web DApp. Browser akan mengubah file menjadi buffer. Lalu Web DApp akan mengirimkan file ke IPFS melalui IPFS-api. Ipfs-api akan memproses pengunggahan *file* menuju IPFS. Pada saat proses pengiriman data melalui IPFS, konfigurasi IPFS menggunakan port 4001 pada saat proses pengiriman data. IPFS akan mengecek integritas dari metadata file dan mengembalikan hash dari file oleh IPNS. Jika data yang masuk dalam jaringan IPFS terbukti orisinil, proses menyimpan file dilakukan oleh Bitswap dengan mengirimkan beberapa blok ke sejumlah *node* yang terhubung. IPFS akan mengirimkan hash dari file ke web DApp dan Ethereum melalui IPFS-api. Sebelum menjalankan smart contract Ethereum akan meminta konfirmasi transaksi melalui Metamask. User akan memberikan konfirmasi dan Metamask akan meneruskan kepada Ethereum. Dengan adanya persetujuan user, smart contract akan dieksekusi oleh Ethereum dengan input data berupa hash IPFS. Transaksi smart contract yang memiliki data hash file akan disimpan di Blockchain. Blockchain akan memberikan transaction recipt ke web DApp melalui web3. Transaction receipt adalah informasi transaksi smart contract seperti hash transaksi, nonce, block hash, block number, gas used, gas price, input dan lain-lain.



Gambar 1. Arsitektur Sistem

4. Metode Penelitian



Gambar 2. Metode Penelitian

Ada 3 tahap dalam penelitian ini yaitu tahap analisis, tahap desain, tahap implementasi dan tahap pengujian. Tahap analisis adalah tahap analisis hardware dan software yang akan digunakan untuk membangun web DApp yang mengintegrasikan IPFS pada smart contract Ethereum. Tahap desain yaitu menggambarkan hubungan antar aplikasi dan alur kerja web DApp yang mengintegrasikan IPFS pada *Smart Contract Ethereum*.

Tahap implementasi yaitu tahap membangun sebuah DApp berbasis *web* yang mengintegrasikan IPFS pada *Smart contract* Ethereums sesuai dengan arsitektur sistem yang telah didesain. Web DApp dibangun diatas sistem operasi Linux Ubuntu 16.04 LTS. Langkah pertama dalam membangun sistem yaitu membuat akun Metamask untuk mendapatkan dompet dan memanajemen transaksi dari Ethereum Blockchain. Kemudian membuat dan deploy contract menggunakan Remix dari akun Metamask. Remix adalah aplikasi *web* bersifat *open source* yang membantu dalam menulis *smart contract* dalam bahasa pemograman Solidity. Langkah berikutnya yaitu membangun web DApp. Adapun aplikasi yang membantu dalam mengembangkan web DApp yaitu Node.js, dan dependency NPM seperti create-react-app, react-bootstrap, fs-extra, ipfs-api dan web3. Kemudian mengintegrasikan web DApp dengan IPFS dan Smart Contract melalui ipfs-api dan web3.

Tahap pengujian dilakukan terhadap tiga laptop yang terhubung dan menjalankan web DApp yang terintegrasi dengan IPFS dan Smart Contract Ethereum melalui proses unggah file. Tujuan dilakukan pengujian yaitu untuk menganalisa pengaruh ukuran file yang diunggah dan jumlah interaksi node pada pemakaian memori saat menjalankan web DApp. Ada tiga skenario pengujian dalam penelitian ini yaitu (1) satu node mengunggah file melalui web DApp, (2) dua node menggunggah file melalui web DApp dalam waktu bersamaan dan (3) tiga node menggunggah file melalui web DApp dalam waktu bersamaan. Pemakaian memori ketiga skenario merupakan hasil kalkulasi dari pemakaian memori seluruh aplikasi yaitu Metamask, Web DApp, IPFS, NPM, dan Node.js. Ukuran pemakaian memori didapatkan dengan memonitoring pemakaian memori melalui aplikasi Htop.

5. Hasil dan Pembahasan

Berikut ini adalah penyajian hasil pengujian tiga skenario penelitian. Skenario pertama adalah satu node mengunggah file melalui web DApp. Skenario kedua adalah node menggunggah file melalui web DApp dalam waktu bersamaan. Skenario ketiga adalah tiga node menggunggah file melalui web DApp dalam waktu bersamaan. Seluruh hasil pengujian diukur dengan besaran satuan *megabyte*.

Tabel 1. Pemakaian Memori Skenario 1 (1 Node)

Proses / Beban	Meta-mask	*Web*DApp	IPFS	NPM	*Node*.js	Total
idle	285	128	34	41	302	791
10MB	335	176	53	41	302	907
50MB	364	205	126	41	303	1041
100MB	392	237	241	41	303	1215
200MB	406	331	376	41	303	1458
300MB	420	429	446	41	303	1639
400MB	427	521	512	41	303	1805
500MB	473	611	545	41	303	1973
600MB	492	707	545	41	303	2089
700MB	527	807	545	41	303	2224
800MB	530	900	545	41	303	2320
900MB	557	997	545	41	303	2443
1GB	567	1084	536	41	303	2532
1.1GB	585	1187	569	41	304	2686
1.2GB	617	1209	588	41	303	2758
1.3GB	650	1381	606	41	304	2982
1.4GB	665	1473	617	41	304	3101
Rata-Rata	488	728	437	41	303	1998

Tabel 2. Pemakaian Memori Skenario 2 (2 Node)

Proses / Beban	Meta-mask	*Web*DApp	IPFS	NPM	*Node*.js	Total
idle	295	126	45	41	305	813
10MB	328	145	63	41	306	883
50MB	354	185	136	41	305	1021
100MB	369	238	234	41	306	1188
200MB	409	327	323	41	305	1406
300MB	411	428	360	41	306	1547
400MB	434	522	360	41	305	1663
500MB	465	619	408	41	306	1840
600MB	482	709	415	41	305	1953
700MB	492	807	412	41	306	2058
800MB	522	902	415	41	306	2186
900MB	545	996	415	41	306	2303
1GB	559	1090	405	41	306	2402
1.1GB	581	1187	406	41	306	2521
1.2GB	588	1281	406	41	306	2622
1.3GB	613	1377	462	41	306	2799
1.4GB	647	1484	470	41	306	2948
Rata-Rata	476	731	337	41	306	1891

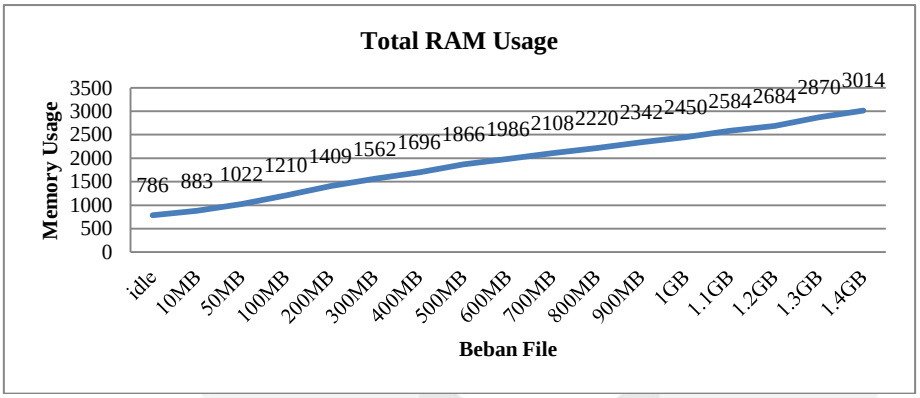
Tabel 3. Pemakaian Memori Skenario 3 (3 Node)

Proses / Beban	Meta-mask	*Web*DApp	IPFS	NPM	*Node*.js	Total
idle	210	126	33	41	310	720
10MB	249	146	46	41	310	792
50MB	252	198	132	41	309	933
100MB	277	297	253	41	310	1178
200MB	305	331	287	41	309	1273
300MB	327	430	287	41	310	1395
400MB	356	519	287	41	310	1513
500MB	415	620	337	41	310	1723

600MB	443	715	345	41	310	1854
700MB	474	805	345	41	310	1975
800MB	494	895	345	41	310	2085
900MB	517	993	364	41	310	2225
1GB	543	1089	401	41	310	2384
1.1GB	581	1184	410	41	310	2527
1.2GB	609	1281	426	41	310	2667
1.3GB	636	1374	463	41	310	2824
1.4GB	699	1468	476	41	310	2995
Rata-Rata	434	733	308	41	310	1827

Dari ketiga skenario terjadi kenaikan pemakaian RAM pada setiap pengunggahan file dengan ukuran besaran file dari 10 MB, 50 MB, 100MB hingga 1.4 GB pada setiap proses. Kenaikan pemakaian RAM pada DApp dan IPFS dipengaruhi oleh besaran ukuran *file*. Pada DApp semakin besar ukuran *file*, semakain besar memori yang dibutuhkan untuk menjalankan seluruh proses kerja dari DApp. Subproses dari DApp yang paling memakan banyak sumber daya yaitu proses convert *file* yang akan dikirim menjadi *buffer*. Pada IPFS, semakin besar ukuran *file* yang dikirim, maka semakin banyak RAM yang dibutuhkan oleh IPFS untuk melakukan seluruh proses seperti pengecekan integritas pada metadata *file* yang akan disimpan, menyimpan *file* pada *block* IPFS, menghasilkan *hash* dari *file* dan memberikan *hash file* kepada DApp.

Sedangkan kenaikan pada Metamask, NPM dan Node.js tidak dipengaruhi oleh besaran ukuran *file* yang diunggah melalui web DApp. Kenaikan pemakaian RAM pada Metamask disebabkan oleh kuantitas transaksi yang semakin banyak. Dengan bertambahnya transaksi pada Metamask, maka data transaksi *smart contract* yang dikelola dan disimpan Metamask juga bertambah banyak. Besaran ukuran *file* tidak berpengaruh terhadap NPM. NPM merupakan package manager untuk membangun *interface* dari DApp. Sehingga ketika melakukan penggunggahan *file*, tidak terjadi perubahan pemakaian memori oleh NPM. Besaran ukuran *file* juga tidak berpengaruh terhadap *Node*.js. Node.js adalah runtime enviroment untuk NPM. Sehingga ketika melakukan penggunggahan *file*, tidak terjadi perubahan pemakaian memori oleh *Node*.js.



Gambar 3. Hasil Total Pemakaian Memori 3 Skenario

Gambar diatas merupakan rata-rata total pemakaian setiap pengujian pengunggahan file dari 10MB-1.4GB. Dari hasil pemakaian memori diatas dapat dilakukan *forecasting* menggunakan metode Trend Least Square.

Tabel 4. Tahap Forecasting (Iskandar, 2014)

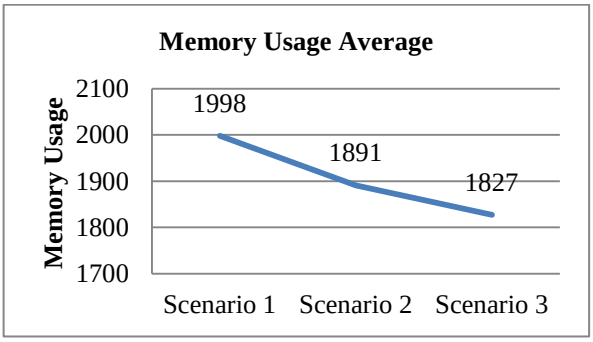
Satuan MB				
N	**y**	**x**	**x.y**	**x2**
100 MB	1194	-13	-15520.96	169
200 MB	1379	-11	-15166.38	121
300 MB	1527	-9	-13740.97	81
400 MB	1660	-7	-11620.84	49
500 MB	1845	-5	-9225.41	25
600 MB	1965	-3	-5895.82	9
700 MB	2086	-1	-2085.68	1
800 MB	2197	1	2196.74	1
900 MB	2324	3	6971.13	9
1 GB	2439	5	12195.86	25
1.1 GB	2578	7	18044.85	49
1.2 GB	2682	9	24141.52	81

1.3 GB	2868	11	31552.97	121
1.4 GB	3014	13	39188.44	169
14	29758	0	61035.44	910

Dengan persamaan trend : Y =a + bx, setelah dilakukan perhitungan, didapatkan nilai a = 2125.6 dan b = 67.07, sehingga menghasilkan persamaan trend pada pemakaian RAM yaitu: y = 2125.6 + 67.07x

Tabel 5. Hasil Forecasting

Satuan MB	
Beban File (X)	**RAM Usage (Y)**
1500	3132
1600	3266
1700	3386
1800	3518
1900	3643
2000	3774
2100	3913
2200	4046
2300	4183
2400	4316
2500	4448
2600	4581
2700	4705
2800	4836
2900	4971
3000	5103
3100	5238
3200	5370
3300	5503
3400	5635
3500	5766
3600	5898
3700	6029
3800	6161
3900	6294
4000	6426
4100	6559
4200	6692
4300	6823
4400	6956
4500	7087
4600	7219
4700	7351
4800	7484
4900	7616
5000	7748



Gambar 4. Perbandingan Total Pemakaian Memori 3 Skenario

Skenario pertama adalah satu node mengunggah file melalui web DApp. Rata-rata dari total pemakaian RAM seluruh pengujian dari 10MB-1.4GB pada skenario pertama yaitu 1998 MB. Skenario kedua adalah node menggunggah file melalui web DApp dalam waktu bersamaan. Rata-rata dari total pemakaian RAM seluruh pengujian dari 10MB-1.4GB pada skenario kedua yaitu 1891 MB. Skenario ketiga adalah tiga node menggunggah file melalui web DApp dalam waktu bersamaan. Rata-rata dari total pemakaian RAM seluruh pengujian dari 10MB-1.4GB pada skenario ketiga yaitu 1827 MB. Dengan interaksi yang semakin banyak pada skenario terjadi penurunan penurunan pemakaian RAM. Oleh karena itu dapat disimpulkan, semakin banyak interaksi *node* yang terjadi maka semakin efesien kinerja dari web DApp yang mengintegrasikan IPFS pada smart contract Ethereum karena pemakaian RAM semakin sedikit.

6. Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, untuk menggunggah *file* hingga 2 GB pada *web* DApp yang mengintegrasikan IPFS pada *smart contract* Ethereum membutuhkan memori dengan spesifikasi RAM miminum yaitu 4 GB. Aplikasi yang paling banyak menghabiskan memori untuk menjalankan *web* DApp yang mengintegrasikan IPFS pada *smart contract* Ethereum adalah *web* DApp itu sendiri yaitu sekitar 732 MB. Kenaikan pemakaian RAM pada setiap proses unggah *file* melalui DApp dipengaruhi oleh besaran ukuran *file* dan intensitas interaksi yang dilakukan oleh *node* dalam jaringan. Efesiensi kinerja web DApp yang mengintegrasikan IPFS pada smart contract Ethereum didapatkan dengan ukuran *file* yang kecil dan jumlah interaksi *node* yang banyak.

Daftar Pustaka:

[1] Atzei N., Bartoletti M., & Cimoli, T., 2017, A Survey of Attacks on Ethereum Smart Contracts, *International Conference on Principles of Security and Trust*, 164-186.

[2] Bennet, J., 2014, IPFS - Content Addressed, Versioned, P2P File System(DRAFT 3), *ArXiv.*

[3] Bryant, R.E & O'Hallaron, D.R. "*Computer Systems: A ProgRAMmer's Perspective, Second Edition",* Pearson, United States of America, 2013

[4] Chen, Yongle, dkk., 2017, An Improved P2P File System Scheme based on IPFS and Blockchain, *IEEE International Conference on Big Data*.

[5] Dannen, C., 2017 , *Introducing Ethereum and Solidity:Foundations of Cryptocurrency and Blockchain ProgRAMming for Beginners*, New York, Apress. doi:10.1007/978-1-4842-2535-6_11

[6] Ethos. 2018. What is Ethereum Gas ? , https://www.ethos.io/what-is-ethereum-gas/, diakses 28 Juni 2019.

[7] Grech, A. dan Camilleri, A. F. "*Blockchain in Education, dalam* Inamorato dos Santos, A. (ed.)", 2017, EUR 28778 EN; doi:10.2760/60649

[8] Kenneth, H., 2018. *Ethereum Test Network* , [Online] Available at: https://medium.com/coinmonks/ ethereum-test-network-21baa86072fa [Accessed 17 June 2019].

[9] Oscar , 2018, AI Smart Contract – The Past, Present, and Future, https://hackernoon.com/ai-smart-contracts-the-past-present-and-future-625d3416807b diakses 28 Juni 2019

[10] Parizi, dkk. "Empirical Vulnerability Analysis of Automated Smart contracts Security Testing on Blockchains". Centre for Advanced Studies Conference, 2018.

[11] Pratama, I. P. A. E, 2014, *Handbook Jaringan Komputer*,Bandung, Informatika.

[12] Rajalakshmi, A., dkk, 2018. A Blockchain and IPFS based Framework for Secure Research Record Keeping, *International Journal of Pure and Applied Mathematics* 119:15 1437-1442

[13] Sinha, P. dan Kaul, A., 2018, Decentralized KYC System, *International Research Journal of Engineering and Technology (IRJET)*, 5:8 1209-1210.

[14] Wang, S., dkk. "A Blockchain-Based Framework for Data Sharing with Fine-grained Access Control in Decentralized Storage Systems". Jurnal IEEE Access,2018, Vol. 4, pp. 1-5. doi:10.1109/access.2018.2851611