Uma Análise Experimental Sistemática do Envelhecimento e Rejuvenescimento da Plataforma Docker

Lucas Santos[1], Laécio Rodrigues[1], Matheus Torquato[2], Francisco Airton Silva[1]

[1] Universidade Federal do Piauí, Campus Picos (UFPI)
[2] Instituto Federal de Alagoas, Campus Arapiraca (IFAL)
Brasil

{vinicius.lucas, laecio8andrade, faps}@ufpi.edu.br

matheus.torquato@ifal.edu.br

Abstract. *Software aging has been a subject explored for almost thirty years. Even with so many years of scientific research, new investigations will be needed due to new technologies. Studies show that containers are lighter than virtual machines, however, running many containers can lead to aging. This paper presents a study of aging and rejuvenation of the Docker platform. An experiment was carried out for thirty days, which indicated different levels of aging by varying hardware capacity. We have applied an approach called SWARE. SWARE facilitates the detection of aging signs and rejuvenation effectiveness in a single experiment.*

Resumo. *Envelhecimento de software é um assunto explorado a praticamente trinta anos. Mesmo com tantos anos de pesquisa científica, novas investigações serão necessárias devido às peculiaridades de novas tecnologias. Contêineres são mais leves do que máquinas virtuais, porém, executar muitos contêineres pode gerar envelhecimento. Este artigo apresenta um estudo de envelhecimento e rejuvenescimento da plataforma Docker. Foi realizado um experimento durante trinta dias, que apontou diferentes níveis de envelhecimento variando capacidades de hardware. Foi aplicado uma abordagem chamada SWARE. A SWARE facilita a detecção de indícios de envelhecimento e efetividade do rejuvenescimento num experimento único.*

1. Introdução

Envelhecimento de software (*software aging*) refere-se à degradação dos recursos computacionais diretamente relacionados à execução do software ao longo do tempo [Grottke et al. 2008]. Existem alguns sinais que indicam o envelhecimento de software tais como: esgotamento de recursos do sistema, corrupção de dados e acumulação de erros. Porém, nem todo sistema apresenta tais sinais de forma clara. Um dos efeitos comuns do envelhecimento de software é o vazamento de memória [Liu et al. 2019], [Liu and Meng 2019]. Algumas pesquisas mostraram esse indicador nas seguintes plataformas: servidores web [Grottke et al. 2006], sistemas operacionais [Cotroneo et al. 2010] e middlewares [Carrozza et al. 2010].

O rejuvenescimento de software é uma atividade proativa destinada a reduzir a probabilidade futura de falhas, devido ao envelhecimento. A ideia básica é pausar ou

interromper a execução do software (ex.: reinicialização do sistema operacional), atualizando seu estado interno e remover o acúmulo de efeitos de envelhecimento de software [Huang et al. 1995]. O rejuvenescimento de software pode ser baseado em uma variedade de indicadores, por exemplo, existe a técnica baseada no tempo decorrido desde o último rejuvenescimento [Polinsky et al. 2020] [Levitin et al. 2019] [Bai et al. 2020], e existe também a técnica baseada em um limiar de consumo de recurso [Araujo et al. 2011b] [Chen et al. 2006].

Com o advento da computação em nuvem e a virtualização, aplicativos distribuídos modernos passaram a ser executados em ambientes virtualizados para uma melhor utilização de recursos de hardware. Devido à alta sobrecarga de recursos, aos poucos as máquinas virtuais vêm sendo substituídas por contêineres [Gillani and Lee 2020]. Os contêineres (ex.: Docker[1]) são uma alternativa à virtualização tradicional por se mostrar mais leve. No entanto, os estudos de envelhecimento relacionados ao uso de contêineres são ainda escassos. O presente artigo estende o trabalho de [Torquato and Vieira 2019] que foi o único trabalho com foco em rejuvenescimento da plataforma Docker. Uma limitação desse trabalho foi realizar experimentos com apenas uma configuração de computador (8GB de RAM).

Este artigo apresenta uma análise experimental sistemática do envelhecimento e rejuvenescimento da plataforma Docker. Como uma extensão do trabalho anterior, foram agora avaliados dois aspectos de forma conjunta em três diferentes configurações de computadores (4GB, 8GB e 16GB de RAM). Primeiramente, foram avaliados o envelhecimento por um período sem interrupções, observando o consumo de memória RAM, swap, e CPU. Posteriormente, foi aplicado o método SWARE, direcionado ao rejuvenescimento da plataforma após um período de paralisação da carga de trabalho.

O experimento sem interrupções mostrou que em máquinas mais potentes o consumo de memória RAM tende a ter um crescimento mais linear, mas isso se deve também ao acionamento da memória Swap em computadores menos potentes. O experimento com a metodologia SWARE mostrou que os computadores com menores capacidades apresentaram alto consumo de memória RAM desde o início do experimento e o rejuvenescimento não foi eficaz no computador menos potente. Pôde-se observar também que houve envelhecimento em relação CPU — mesmo que em baixo consumo — e em relação ao processo dockerd, do Docker. Acredita-se que esse é o primeiro trabalho a lidar com envelhecimento da plataforma Docker em múltiplas configurações de computadores e com experimentos sem interrupções. Os métodos e resultados apresentados aqui podem ajudar os administradores da plataforma Docker a elaborar políticas de gerenciamento dos seus ambientes.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta o primeiro estudo, focado no envelhecimento de longo prazo. A Seção 4 apresenta o segundo estudo, focado na paralisação da carga de trabalho e posterior rejuvenescimento do sistema. A Seção 5 traça algumas conclusões e direcionamentos futuros.

[1]https://www.docker.com/

2. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados. A Tabela 1 resume um comparativo dos trabalhos relacionados sob seis aspectos: Plataforma, Métricas, Duração, Rejuvenescimento, Múltiplas Configurações e Sem Interrupção.

Tabela 1. Trabalhos Relacionados

Trabalhos	Plataforma	Métricas	Duração	Rejuven.	Múltiplas Configurações	Sem Interrupção
[Li et al. 2002]	Servidor web Apache	Memória RAM, Swap, Disco, CPU , Processos, Tempo de Resposta	7 dias, 25 dias, 14 dias	Não	Não	Não
[Grottke et al. 2006]	Servidor web Apache	Tempo de Resposta, Memória RAM, Swap	25 dias	Não	Não	Não
[Araujo et al. 2011a]	Eucalyptus Cloud Platform	Memória RAM, Swap, Fragmentação de Memória	10 dias	Sim	Não	Não
[Araujo et al. 2011b]	Eucalyptus Cloud Platform	Memória RAM, Swap, CPU, Disco, Processos	3 dias	Sim	Não	Não
[Meng et al. 2016]	J2EE Application Server	Memória heap da JVM, CPU, Tempo de Resposta, Taxa de Transferência	19.7 horas	Sim	Não	Não
[Melo et al. 2017]	OpenStack Cloud Platform	Memória RAM, Swap, CPU, Disco, Processos	7 dias	Não	Não	Não
[Torquato and Vieira 2019]	Docker	Memória RAM, Swap, CPU	26 dias	Sim	Não	Não
Este Trabalho	Docker	Memória RAM, Swap, CPU	30 dias	Sim	Sim	Sim

O primeiro critério de comparação focou na **Plataforma** avaliada. Os trabalhos relacionados utilizaram diversas plataformas. No entanto, pode-se observar que algumas foram utilizadas por mais de um trabalho, como é o caso do servidor web Apache [Li et al. 2002], [Grottke et al. 2006] e o Eucalyptus [Araujo et al. 2011a], [Araujo et al. 2011b]. Tecnologias relacionadas a computação em nuvem foram amplamente estudadas sobre envelhecimento de software, podendo ser notado que além do Eucalyptus a plataforma OpenStack também foi citada. No entanto, apenas o trabalho de [Torquato and Vieira 2019] fez experimentos com foco na plataforma Docker. Como falado anteriormente, o presente artigo é uma extensão do trabalho de [Torquato and Vieira 2019].

Sobre as **Métricas** observadas referentes ao envelhecimento, destacam-se memória RAM, memória Swap, e CPU. Outras métricas também foram estudadas, mas com menos frequência, como o tempo de resposta [Li et al. 2002], [Grottke et al. 2006], [Meng et al. 2016]. O tempo de resposta é importante ser avaliado quando aspectos como Quality of Experience negativo e questões de indisponibilidade causarem problemas aos usuários. Foi observado também o uso da métrica Processos, que diz respeito a processos secundários observados durante o monitoramento do sistema [Li et al. 2002], [Araujo et al. 2011a], [Melo et al. 2017]. Os processos avaliados nesses três trabalhos referem-se aos processos relacionados ao software avaliado em questão, que se tratando de plataforma de nuvem, existem diversos processos relacionados ao serviço. No entanto no caso do Docker, tanto o presente trabalho quanto o trabalho de

[Torquato and Vieira 2019] observou apenas o processo "dockerd". Por se tratar do processo responsável por gerenciar os contêineres.

A **Duração** dos experimentos também é uma característica importante. Dependendo do tempo de observação pode-se ou não identificar padrões de envelhecimento. O mais curto período foi realizado pelo trabalho de [Araujo et al. 2011a] com apenas três dias de observação. O período mais longo foi aplicado pelo experimento proposto neste trabalho. Porém, os trabalhos de [Torquato and Vieira 2019] e [Grottke et al. 2006] foram bem próximos, com 26 e 25 dias respectivamente. Se o pesquisador está interessado em descobrir o ponto de exaustão dos recursos, é primordial demorar semanas ou até vários meses. Por isso, a carga de trabalho é um ponto igualmente relevante.

O **Rejuvenescimento** de software é uma ação fundamental para trazer de volta a um estado estável um sistema que teve seus recursos exauridos. Alguns trabalhos optaram por não só analisar os efeitos do envelhecimento de software, mas também a eficácia do rejuvenescimento. Apenas três trabalhos não exploraram rejuvenescimento: [Li et al. 2002], [Grottke et al. 2006] e [Melo et al. 2017]. Acredita-se que rejuvenescimento é uma ação que deve ser explorada em toda plataforma que haja envelhecimento.

Outro aspecto analisado foram as **Múltiplas Configurações** de hardware. Nos ambientes de testes, deve-se observar as configurações de hardware e como as mesmas tendem a influenciar no avanço do envelhecimento de software. Este trabalho destaca-se de todos os demais analisados. Isso por ser o único a realizar um estudo experimental de envelhecimento e rejuvenescimento em uma plataforma utilizando mais de uma configuração de hardware no mesmo experimento. Considera-se tal característica uma contribuição pois no trabalho anterior [Torquato and Vieira 2019], o uso de apenas uma configuração de hardware limitou fazer novas descobertas. O presente artigo avança em termos de variação experimental, procurando identificar o impacto do envelhecimento sobre a plataforma Docker em diferentes configurações de hardware.

O último critério a ser observado é se os experimentos ocorreram em um longo prazo **Sem Interrupções**. No trabalho anterior [Torquato and Vieira 2019], teve a execução do experimento interrompida quando alcançou aproximadamente sete dias. No presente artigo foi executada uma abordagem distinta. No primeiro experimento foram observados 30 dias de execução sem interrupções.

3. Experimento de Envelhecimento sem Interrupção

Essa seção apresenta um experimento inicial que teve o objetivo de encontrar evidencias que comprovem o envelhecimento da plataforma por muitos dias, sem interrupção.

3.1. Metodologia

Para o ambiente de testes, foram utilizados três computadores, como forma de obter uma maior gama de dados para serem analisados. As configurações de hardware de cada um dos computadores são apresentadas na Tabela 2. Em todos os computadores foi instalado o sistema operacional Ubuntu Server 18.04.4 LTS. O experimento consistiu em submeter os três computadores a uma constante carga de trabalho utilizando a plataforma *Docker*, consumindo os recursos do sistema, ao mesmo tempo que tais recursos eram monitorados. Todo o experimento foi realizado ao longo de um período de 30 dias consecutivos, e ao seu término os dados foram coletados e analisados.

Tabela 2. Configurações dos Computadores A, B e C

Computador	Processador	Memória RAM	Memória Swap	Memória Disco Rígido
A	Intel i3 de 3.10 GHz	4GB	2GB	500GB
B	Intel i3 de 3.10 GHz	8GB	4GB	500GB
C	Intel Xeon 3.3 GHz	16GB	4GB	1TB

O Docker versão 17.05.0-ce foi instalado nos três computadores. Em um ciclo constante, um conjunto de contêineres foram criados e monitorados. Todos as imagens dos contêineres continham os softwares Owncloud[2] e MySQL[3]. Como forma de simular o uso constante dos computadores, estressar o sistema e acelerar o fenômeno de envelhecimento, foi criada uma carga de trabalho para ser executada igualmente em cada um dos computadores. A carga de trabalho foi desenvolvida em forma de *shell script*, consistindo em uma repetição sucessiva de operações para inicializar e finalizar contêineres Docker com as imagens citadas.

Portanto, a carga de trabalho consiste na criação de 50 contêineres através do comando *docker create*, de dois em dois contêineres. A cada 10 segundos era verificado se o número de contêineres criados era menor do que 50. Caso afirmativo, dois novos contêineres eram instanciados e o processo se repetiu até que a condição anterior foi dada como falsa. Ao completar 50 contêineres, todos estes foram finalizados com o comando *docker stop* e removidos com o comando *docker rm*. Em seguida, ocorre um aguardo de 50 segundos e todo o processo é executado novamente, ininterruptamente. O sistema é monitorado em paralelo à execução da carga de trabalho, para isso são utilizados comandos de um outro *shell script*. Esse script, por sua vez, utilizam chamadas de sistema padrão do próprio Linux, como *free -mh, top* e *grep* para coletar os dados referentes às métricas definidas como indicadores do envelhecimento de software: memória principal de todo o sistema, Swap e CPU.

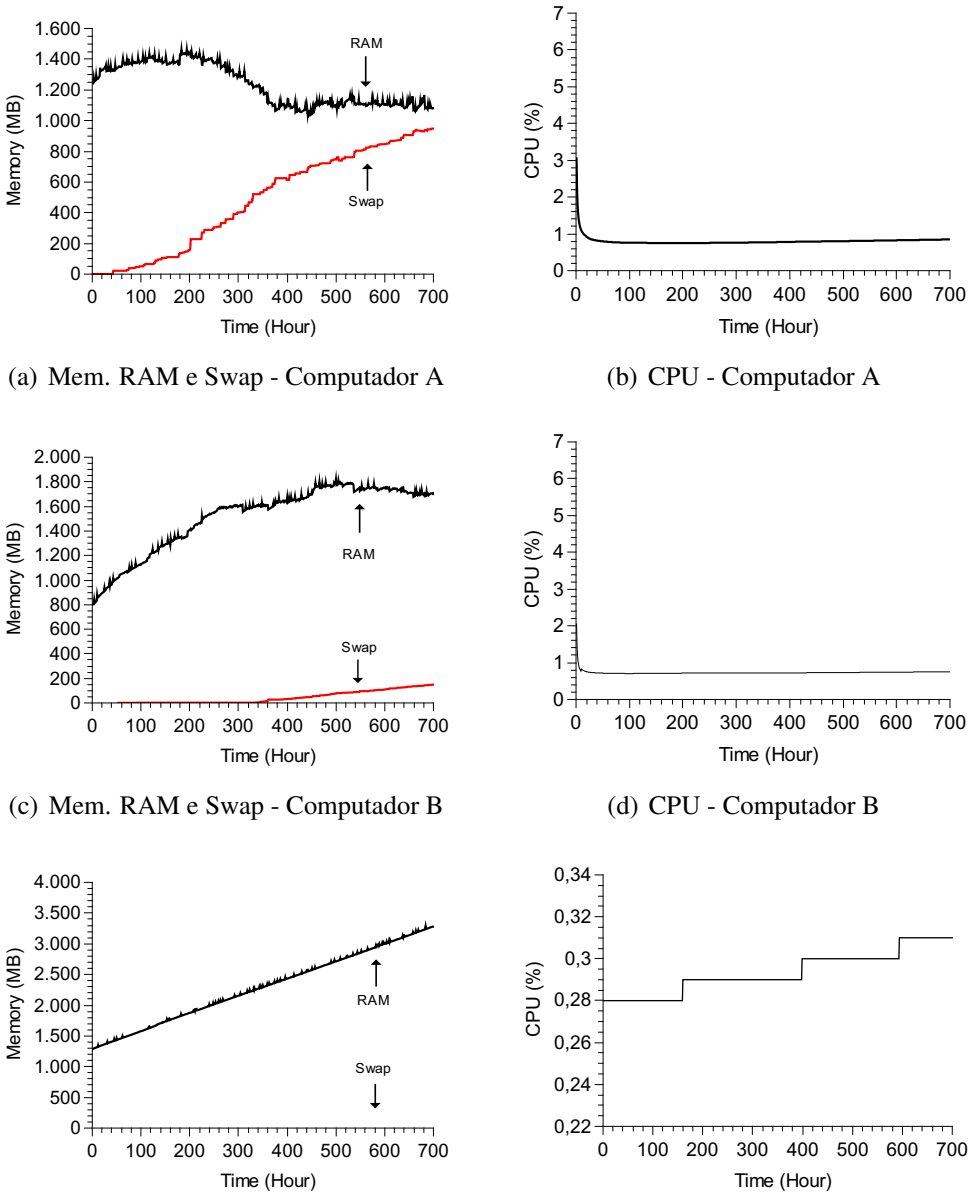
3.2. Resultados

A Figura 1 apresenta os resultados da análise dos dados coletados durante o experimento para os computadores A, B e C, respectivamente. Para cada computador são apresentados dois gráficos, um com RAM + Swap e outro com CPU. Após realizar a análise citada, constataram-se indícios de exaustão dos recursos do sistema e que sugerem envelhecimento de software.

As Figuras 1(a) e 1(b) apresentam, respectivamente a utilização de memória RAM + Swap e percentual de uso da CPU do computador A durante todo o período do experimento. Percebe-se na Figura1(a) que ao final de um pouco mais de 200 horas, o uso de memória RAM atinge um ápice de aproximadamente 1.400MB. Quase nesse mesmo momento, o uso de memória Swap do computador A começa a crescer exponencialmente, enquanto que o uso da memória RAM apenas decai após atingir o seu ápice. Em contrapartida, o percentual da utilização da CPU em nível do usuário (Figura1(b)) não apresentou variações significativas, mantendo-se constante.

As Figuras 1(c) e 1(d) apresentam o resultado para o computador B. O computador B, por possuir uma configuração de hardware mais avançada demorou mais tempo

[2]Owncloud: https://owncloud.org/
[3]MySQL: https://www.mysql.com/



(a) Mem. RAM e Swap - Computador A

(b) CPU - Computador A

(c) Mem. RAM e Swap - Computador B

(d) CPU - Computador B

(e) Mem. RAM e Swap - Computador C

(f) CPU - Computador C

Figura 1. Utilização Geral de Recursos dos Computadores A, B e C

para acionar a memória Swap. A Figura 1(c) revela que o uso de memória RAM cresce constantemente até aproximadamente 300 horas. Tendo seu crescimento reduzido a partir desse ponto até atingir o seu ápice de aproximadamente 1.800MB, pouco depois de 500 horas, decaindo levemente a partir desse ponto. A Figura1(c) também mostra que um pouco depois de 300 horas, a memória Swap é acionada, porém com um baixo crescimento. Coincidindo com o momento em que o uso da memória RAM deixa de ter um crescimento exponencial. Em relação à CPU (Figura 1(d)), da mesma forma que o computador A, o computador B não apresentou variações significativas em relação ao percentual de uso de CPU, mantendo-se constante durante toda a execução.

As Figuras 1(e) e 1(f) apresentam o resultado para o computador C. O compu-

tador C por possuir o hardware mais avançado dentre os três computadores, teve um crescimento constante do consumo de memória RAM, porém sem acionar Swap. Mesmo não acionando Swap este crescimento de memória RAM sugere indicação de envelhecimento. Este crescimento linear é semelhante ao crescimento nos momentos mais iniciais dos computadores A e B. A CPU (Figura 1(f)), apesar de apresentar alteração, tem valores desprezíveis de consumo.

Este experimento sem interrupções foi necessário para observar o comportamento do consumo de recursos ao longo de muitos dias. Apesar de apresentar resultados importantes sobre envelhecimento, não foi aplicado rejuvenescimento. O objetivo da próxima seção é observar este outro aspecto, que é monitorar o consumo de recursos ao parar a geração de carga de trabalho e com aplicação da reinicialização do sistema operacional.

4. Análise de Envelhecimento com a Metodologia SWARE

Esta seção apresenta um segundo experimento de envelhecimento de software na plataforma Docker usando uma metodologia denominada SWARE. O experimento buscou observar como os recursos se comportam quando uma carga de trabalho sintética é cessada após um certo período. Além disso, foi testados o que ocorre ao realizar uma ação de rejuvenescimento. O presente experimento foi executado nos mesmos três computadores do experimento anterior.

4.1. Metodologia

Em [Torquato et al. 2018] foi proposto uma metodologia de teste de envelhecimento de software denominada SWARE, e posteriormente esta metodologia foi aplicada na plataforma Docker [Torquato and Vieira 2019]. O objetivo da metodologia SWARE é analisar os indicadores de envelhecimento de software e observar os efeitos de ações de rejuvenescimento. A abordagem é caracterizada por três fases: **Stress**, **Wait** e **Rejuvenation**.

Na fase de **Stress**, o computador é submetido a uma constante carga de trabalho como forma de exaurir os recursos do sistema e ativar os *bugs* de envelhecimento. Na fase de **Wait** a carga de trabalho é cessada, passando a se analisar as consequências causadas pela fase de Stress. Espera-se que os efeitos do envelhecimento de software acumulados durante a fase de Stress devem permanecer durante a fase de Wait. Na fase de **Rejuvenation**, são aplicadas as técnicas de rejuvenescimento de software. Como ação de rejuvenescimento foi aplicada a reinicialização dos computadores. O objetivo é trazer o sistema de volta a um estado estável e sem o acúmulo de erros, sendo esse considerado um rejuvenescimento eficiente. As Figuras 2a e 2b exemplificam resultados sem envelhecimento e com envelhecimento na fase de Wait. Vale enfatizar que para a metodologia SWARE, os indícios de envelhecimento de software são reconhecidos quando o consumo de recursos na fase Wait permanece constante ou crescente. Na fase de Rejuvenescimento, as Figuras 2c e 2d exemplificam resultados de rejuvenescimento eficiente e rejuvenescimento ineficiente.

Neste trabalho, foi aplicada a metodologia SWARE no mesmo ambiente de testes do experimento anterior (i.e., mesmo aparato computacional e mesmo script para monitoramento). Porém, foram aplicadas alterações na geração de carga. Nesse experimento, para acelerar os resultados, foram instanciados 480 contêineres em cada ciclo, sendo 10 segundos de intervalo a cada instanciação de 16 contêineres e 50 segundos de espera após

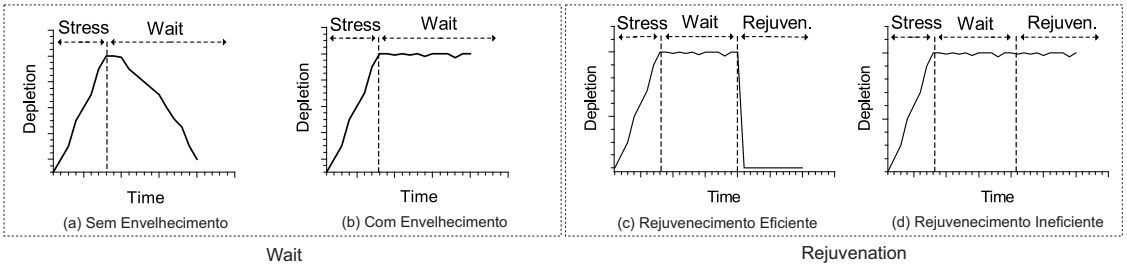


Figura 2. Exemplificação das fases de Wait e Rejuvenation

a exclusão dos contêineres. Como no experimento anterior o tempo de longa duração foi suficiente para entender o comportamento do envelhecimento a longo prazo, agora com a metodologia SWARE, poucos dias foram suficientes para atingir o objetivo.

A fase de Stress durou dois dias. A fase de Wait durou um dia, e a fase de rejuvenescimento durou dezessete horas. Vale ressaltar que durante todo o experimento apenas a plataforma Docker e os scripts estavam sendo executados.

4.2. Resultados

Esta seção apresenta os resultados do experimento usando a metodologia SWARE. Em cada um dos computadores foram observadas as seguintes métricas: Consumo de Memória RAM, Consumo de Memória Swap, Percentual de CPU usada e Percentual de Memória Usada pelo processo dockerd. Cada gráfico possui marcadores para cada fase do experimento. O final da fase de Stress é delimitado com uma linha Azul. Nesse ponto ocorre a parada da geração de carga de trabalho. O final da fase de Wait é demarcado com uma segunda linha azul, sendo o sistema operacional reiniciado a partir desse ponto.

A Figura 3 apresenta os resultados do uso de memória RAM dos computadores A, B e C. Há um crescente consumo de memória RAM durante a fase de Stress nos três computadores. No entanto, enquanto que os computadores A e B (menos potentes) já iniciam o processo com alto consumo ($\approx$1100MB), o computador C inicia a partir de $\approx$200MB. Curiosamente, o pico do consumo do computador C se iguala no final dessa fase com o computador A, enquanto que o computador B se encontra mais alto. Na fase de Wait os três computadores apresentaram indícios de envelhecimento pois o consumo se manteve constante. Os computadores A e C se mantiveram praticamente com mesmo nível de consumo. O computador B se manteve constante, porém com uma leve queda. Quanto ao rejuvenescimento, os computadores B e C tiveram uma queda significativa do consumo. No entanto, no computador A quase se observou diminuição do consumo. Portanto, deve-se atentar à capacidade da máquina para obter resultados relevantes quanto ao rejuvenescimento.

A Figura 4 apresenta os resultados do uso de memória Swap dos computadores A, B e C. Os computadores B e C apresentaram níveis baixíssimos de consumo da memória Swap. O computador B durante a fase de Stress, alcançou um pico máximo de apenas $\approx$2MB, embora tenha um crescimento ínfimo, mostrou um consumo constante durante a fase de Wait. A memória Swap não chegou a ser acionada pelo sistema no computador C, mantendo-se exatamente com 0MB de memória durante todo o experimento. O computador A apresentou um consumo crescente de memória Swap durante a fase de Stress, alcançando um pico de $\approx$320MB, além de manter-se constantemente alto durante toda a

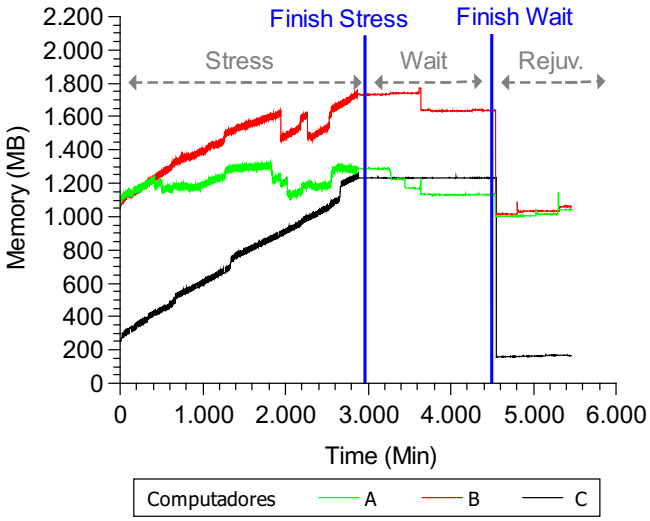


Figura 3. Mem. RAM Computadores A, B e C

fase de Wait. Nesse caso, foi no computador A que se pôde observar um envelhecimento mais significativo considerando a fase de Stress e Wait, por ser o computador de menor capacidade. Em relação ao rejuvenescimento, o computador A apresentou os níveis de consumo de memória Swap diminuídos drasticamente. Os computadores B e C já estavam com níveis muito baixos de consumo quando se acionou o rejuvenescimento, que por sua vez não foi tão impactante.

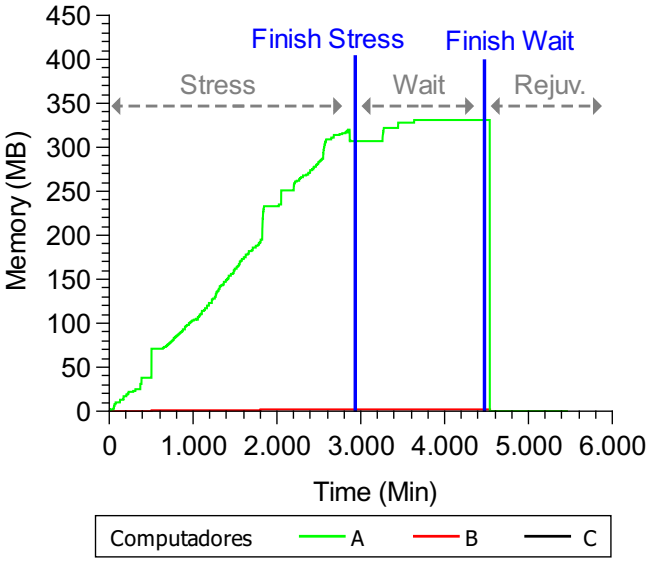


Figura 4. Mem. Swap Computadores A, B e C

A Figura 5 apresenta o percentual do uso total de CPU dos computadores A, B e C. Os computadores A e B apresentaram uso de CPU bastante semelhantes. No computador A o uso de CPU manteve-se baixo durante a fase de Stress, alcançando um pico máximo de ≈8%. O computador B alcançou um consumo máximo de ≈4% durante a fase de Stress. Observa-se que na fase de Wait o percentual de ambos os computadores se manteve levemente constante, ou seja, havendo um certo grau de envelhecimento pós parada. Durante a fase Wait, é suposto que o sistema esteja ocioso. Logo, o uso do proces-

sador indica que ainda existem (quando não deveriam) tarefas a consumir recursos. Para o computador C os resultados do uso de CPU novamente não se mostraram significativos. Iniciando com ≈1% de uso e se mantendo baixo e constante na maior parte do tempo e com um leve decaimento durante a fase de Wait. O rejuvenescimento mostrou-se eficiente em relação ao uso de CPU nos três computadores, apresentando apenas pequenos picos de uso durante a inicialização.

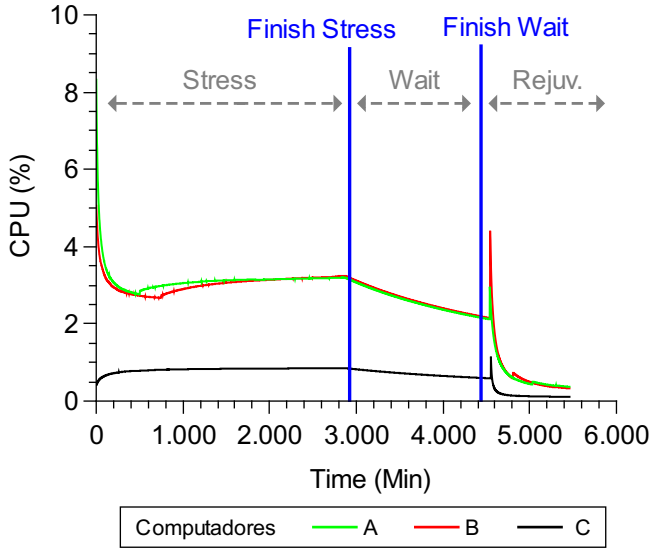


Figura 5. CPU dos Computadores A, B e C

A Figura 6 apresenta o percentual do uso de memória do processo *dockerd* nos computadores A, B e C respectivamente. No computador A o percentual do uso de memória do processo *dockerd* mostrou-se crescente durante a fase de Stress, alcançando um pico de ≈7%, além de manter um uso constante durante a fase de Wait. No computador B os resultados apresentaram-se semelhantes ao computador anterior somente na fase inicial, na fase de Wait o consumo se manteve alto mais alto. No computador C, o percentual do uso de memória do processo *dockerd* foi o menor dos três computadores. Houve um uso crescente na fase de Stress, alcançando um ápice de ≈5% e um uso constante durante a fase de Wait, repetindo o comportamento dos casos anteriores, porém de forma mais baixa. O rejuvenescimento mostrou-se eficiente em todos os três computadores, fazendo com que o percentual do uso de memória do processo *dockerd* caísse drasticamente para ≈0%.

Os resultados apresentados nessa seção corroboram os resultados obtidos em [Torquato and Vieira 2019]. Vale salientar que a fase de Stress visa apenas forçar o sistema a consumir mais recursos num curto espaço de tempo. Assim, espera-se que os possíveis *bugs* de envelhecimento de software sejam ativados precocemente. Nos resultados obtidos, percebe-se que após apenas dois dias de estresse, os efeitos de envelhecimento já são perceptíveis. Isso sugere que a ativação dos *bugs* de envelhecimento podem ocorrer em situações normais de consumo de recursos. Por exemplo, o consumo de CPU na fase de estresse não ultrapassou 6%. Porém, a fase de espera mostra que alguns efeitos foram acumulados após o estresse.

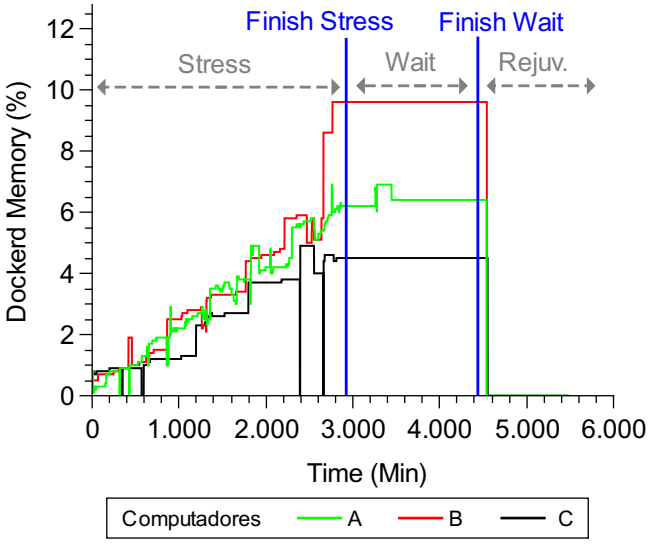


Figura 6. Processo *dockerd* Computadores A, B e C

5. Considerações Finais

Este artigo apresentou uma análise experimental sistemática do envelhecimento e rejuvenescimento da plataforma Docker. Na primeira etapa foi executado um experimento de duração de trinta dias visando a observação do envelhecimento com execução da plataforma sem interrupções. Um segundo experimento, usando a metodologia SWARE, foi realizado com uma atenção maior na etapa após a estagnação da carga de trabalho e ação de rejuvenescimento. Os resultados do experimento de longo prazo apontaram o envelhecimento da plataforma. No entanto, o uso da memória Swap foi representativo apenas quando o recurso computacional da máquina era baixo. Os resultados com abordagem SWARE também apontaram indícios de envelhecimento. Apenas no computador de menor recurso houve uma queda significativa da memória RAM ao parar a carga de trabalho, pois neste caso a memória Swap estava em alto consumo (350MB). Como trabalho futuro, pretende-se fazer uma terceira análise focada na predição do comportamento da plataforma Docker usando técnicas de séries temporais e aprendizagem de máquina.

Referências

Araujo, J., Matos, R., Maciel, P., Matias, R., and Beicker, I. (2011a). Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure. In *Proceedings of the Middleware 2011 Industry Track Workshop*, pages 1–7.

Araujo, J., Matos, R., Maciel, P., Vieira, F., Matias, R., and Trivedi, K. S. (2011b). Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds. In *2011 IEEE Third International Workshop on Software Aging and Rejuvenation*, pages 38–43. IEEE.

Bai, J., Chang, X., Machida, F., Trivedi, K. S., and Han, Z. (2020). Analyzing software rejuvenation techniques in a virtualized system: Service provider and user views. *IEEE Access*, 8:6448–6459.

Carrozza, G., Cotroneo, D., Natella, R., Pecchia, A., and Russo, S. (2010). Memory leak analysis of mission-critical middleware. *J. Syst. Softw.*, 83(9):1556–1567.

Chen, X.-E., Quan, Q., Jia, Y.-F., and Cai, K.-Y. (2006). A threshold autoregressive model for software aging. In *2006 Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06)*, pages 34–40. IEEE.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S. (2010). Software aging analysis of the linux operating system. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 71–80. IEEE.

Gillani, K. and Lee, J.-H. (2020). Comparison of linux virtual machines and containers for a service migration in 5g multi-access edge computing. *ICT Express*, 6(1):1–2.

Grottke, M., Li, L., Vaidyanathan, K., and Trivedi, K. S. (2006). Analysis of software aging in a web server. *IEEE Transactions on reliability*, 55(3):411–420.

Grottke, M., Matias, R., and Trivedi, K. S. (2008). The fundamentals of software aging. In *2008 IEEE International conference on software reliability engineering workshops (ISSRE Wksp)*, pages 1–6. Ieee.

Huang, Y., Kintala, C., Kolettis, N., and Fulton, N. D. (1995). Software rejuvenation: Analysis, module and applications. In *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers*, pages 381–390. IEEE.

Levitin, G., Xing, L., and Huang, H.-Z. (2019). Optimization of partial software rejuvenation policy. *Reliability Engineering & System Safety*, 188:289–296.

Li, L., Vaidyanathan, K., and Trivedi, K. S. (2002). An approach for estimation of software aging in a web server. In *Proceedings International Symposium on Empirical Software Engineering*, pages 91–100. IEEE.

Liu, J. and Meng, L. (2019). Integrating artificial bee colony algorithm and bp neural network for software aging prediction in iot environment. *IEEE Access*, 7:32941–32948.

Liu, J., Tan, X., and Wang, Y. (2019). Cssap: Software aging prediction for cloud services based on arima-lstm hybrid model. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 283–290. IEEE.

Melo, C., Araujo, J., Alves, V., and Maciel, P. R. M. (2017). Investigation of software aging effects on the openstack cloud computing platform. *JSW*, 12(2):125–137.

Meng, H., Hei, X., Zhang, J., Liu, J., and Sui, L. (2016). Software aging and rejuvenation in a j2ee application server. *Quality and Reliability Engineering International*, 32(1):89–97.

Polinsky, I., Martin, K., Enck, W., and Reiter, M. K. (2020). nm-variant systems: Adversarial-resistant software rejuvenation for cloud-based web applications. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 235–246.

Torquato, M., Araujo, J., Umesh, I., and Maciel, P. (2018). Sware: A methodology for software aging and rejuvenation experiments. *Journal of Information Systems Engineering & Management*, 3(2):15.

Torquato, M. and Vieira, M. (2019). An experimental study of software aging and rejuvenation in dockerd. In *2019 15th European Dependable Computing Conference (EDCC)*, pages 1–6. IEEE.