Um Sistema de Monitoramento de Anomalias em Containers de Docker Baseado em Floresta de Isolamento Otimizado

Zhuping Zou, Yulai Xie, *Membro, IEEE*, Kai Huang, Gongming Xu, Dan Feng, *Membro, IEEE,* e Darrell Long, *Fellow, IEEE*

A virtualização com base em contêineres abstratos se tornou gradualmente uma solução principal nos ambientes atuais de computação em nuvem. A detecção e análise de anomalias em containers representam um grande desafio para os fornecedores e usuários de cloud computing. Este artigo propõe um sistema online de detecção de anomalias em contêineres, monitorando e analisando as métricas de recursos multidimensionais dos contêineres, com base no algoritmo florestal de isolamento otimizado. Para melhorar a precisão da detecção, ele atribui a cada métrica de recurso um peso e muda a seleção aleatória de recursos no algoritmo florestal de isolamento para a seleção ponderada de recursos de acordo com o viés de recursos do contêiner. Além disso, o algoritmo
ele pode identificar métricas anormais de recursos e ajustar automaticamente o período de monitoramento para reduzir o atraso de monitoramento e a sobrecarga do sistema. Além disso, ele pode localizar a causa das anomalias através da análise e exploração do registro do contêiner. Os resultados experimentais demonstram o desempenho e eficiência do sistema na detecção das anomalias típicas em contêineres, tanto em ambientes simulados quanto em ambientes de nuvens reais.

Índice Termos-Docker container, monitoramento de anomalias, florestas de isolamento, análise de toras

Ç

1 Introdução

With a popularidade das plataformas de computação em nuvem, cada vez mais empresas têm seus próprios dados centers, prestando serviços a clientes com necessidades diferentes. Uma das tecnologias-chave no centro de dados é a virtualização. O container portuário [1], como uma nova tecnologia de virtualização, tem muitas vantagens atraentes, tais como fácil de implantar e rápido start-up. Assim, tornou-se rapidamente o querido das grandes empresas (por exemplo, Amazon [2], IBM [3] e Oracle [4]).

Entretanto, com a crescente aplicação em larga escala de grupos de contêineres, a questão da segurança e estabilidade dos contêineres também tem chamado cada vez mais a atenção. Por exemplo, o colapso da Amazon Cloud que se baseia em cluster de contêineres e máquinas virtuais levou à invalidação de milhares de websites e aplicativos [5]. Portanto, é crucial detectar anormalidades no contêiner de forma oportuna para garantir a qualidade do serviço da nuvem.

Como os recipientes continuam a subir e descer, um dos desafios é como monitorar múltiplos recursos ao mesmo tempo em um ambiente dinâmico com uma baixa sobre-cabeça. Métodos baseados em regras [6], [7], [8] detectam anormalidades ao estabelecer um limiar para cada métrica. Eles assumem que

• *Z. Zou, Y. Xie, K. Huang, G. Xu, e D. Feng estão na Escola de Informática, Laboratório Nacional de Optoeletrônica de Wuhan, Universidade de Ciência e Tecnologia de Huazhong, Wuhan 430074, R.P. China.*
E-mail: {zouzhup, xugongming38}@gmail.com, {ylxie, keithkhuang, dfeng}@hust.edu.cn.

• *D. Long está na Jack Baskin School of Engineering, University of California, Santa Cruz, CA 95064 USA. E-mail: darrell@ucsc.edu.*

Manuscrito recebido em 19 de agosto de 2018; revisado em 24 de setembro de 2018; aceito em 11 de agosto de 2019. Data de publicação 20 de agosto de 2019; data da versão atual 8 de março de 2022. (Autor correspondente: Yulai Xie.)

Recomendado para aceitação por M. Qiu. Identificador de Objeto Digital nº. 10.1109/TCC.2019.2935724

Apenas um contêiner está funcionando no host no início e estabelece um limite fixo para cada métrica de recursos do contêiner. Quando outro recipiente é criado com uma prioridade de recursos, o limite de recursos original do primeiro recipiente é ajustado de acordo com o uso de recursos do segundo recipiente. Este ajuste se torna impraticável quando existem inúmeros e dinâmicos contêineres de cang- ing. O método baseado em estatísticas [9] pressupõe que os dados obedecem a alguns modelos de distribuição padrão e descobre os valores anômalos que se desviam da distribuição. Como a maioria dos modelos é baseada em suposições univariadas, eles não são aplicáveis a dados multidimensionais. A fim de resolver os problemas acima mencionados, a comunidade acadêmica propôs um método baseado em densidade como o Fator Local Outlier (LOF) [10] e a Detecção de Outlier com Base em Ângulo (ABOD) [11]. Eles identificam aberrações estimando a densidade dos dados locais ou calculando a mudança de ângulo. Entretanto, ambos incorrem em uma grande sobrecarga de cálculo quando o tamanho da amostra de dados é grande.

Os sistemas de monitoramento existentes (por exemplo, Ganglia [6], Nagios [8], Akshay [12], cAdviosr [13]) geralmente adotam um período fixo de mono-torção para consultar a anormalidade do sistema. Quando o período de monitoramento é muito pequeno, o sistema de monitoramento pode localizar rapidamente as anormalidades. No entanto, isto resulta em uma sobrecarga enorme do sistema quando há muitos objetos de monitoramento. Quando o período de monitoramento é grande, o atraso do monitoramento também aumentará. Portanto, é necessário adotar um período de monitoramento adequado de acordo com o estado de funcionamento do sistema.

Quando ocorre uma exceção em um recipiente, geralmente causa uma mudança no uso do recurso do recipiente. Por exemplo, um loop infinito em um programa em execução pode comer todo o recurso da CPU, e um vazamento de memória fará com que o uso da memória se torne maior. Portanto, é necessário identificar o

anomalia através do monitoramento da métrica dos recursos de contêineres. Este documento propõe um sistema de monitoramento de anomalias de contêineres baseado em florestas de isolamento otimizadas. O sistema primeiro obtém cada taxa de utilização de recursos de cada contêiner na máquina hospedeira de forma não intrusiva. Quando dados suficientes de monitoramento são coletados, o valor da anomalia de cada dado de monitoramento é calculado usando a floresta de isolamento otimizada, que leva em conta as características da carga de trabalho da aplicação do contêiner. Especificamente, o sistema atribui um peso a cada métrica de recurso. Se uma aplicação de contêiner depende muito de uma métrica de recurso (por exemplo, a aplicação intensiva de IO depende da taxa de leitura/gravação em disco mais do que a largura de banda da rede), o sistema atribuirá um grande valor a essa métrica de recurso. De forma correspondente, mudamos a seleção aleatória de recursos para a seleção ponderada de recursos ao escolher um fea- ture dos dados para dividir o conjunto de dados no algoritmo de isolamento florestal. Assim, se um recurso métrico com grande peso estiver em um estado anormal, será mais fácil escolhê-lo como a característica para dividir o conjunto de dados. Portanto, a anomalia pode ser identificada com mais precisão. Quando o valor da anomalia de um dado monitorado excede um limite pré-definido, uma anomalia é determinada. Então, o sistema identifica a causa da anomalia através da análise dos registros do contêiner. Ao mesmo tempo, o sistema pode aumentar ou diminuir o período de monitoração de acordo com o grau de anomalias. Assim, ele pode reduzir significativamente o atraso do alarme e monitorar as despesas gerais.

As contribuições deste documento são as seguintes:

- Projetamos um sistema de monitoramento de anomalias de contêineres portuários que pode monitorar a métrica de recursos multidimensionais, ajustar automaticamente o período de monitoramento e analisar a causa das anomalias.
- Propomos um algoritmo florestal de isolamento otimizado que estabelece pesos para diferentes métricas de recursos e pode localizar a métrica anômala de recursos, levando em conta o tipo de carga de trabalho de aplicação do contêiner.
- Implementamos tanto o sistema quanto algo- rithm e os avaliamos tanto em ambientes simulados como em nuvens comerciais reais (AWS) em uma grande variedade de casos de anomalias em termos de detecção de acesso, atraso de monitoramento e análise de log.

2 Antecedentes e Trabalho Relacionado

Nesta seção, primeiramente descrevemos a tecnologia de fundo do Docker e da floresta de isolamento. Em seguida, elaboramos o trabalho relacionado ao sistema de monitoramento e aos métodos de detecção de anomalias.

2.1 Tecnologia Docker

Docker é uma solução de virtualização leve que é essencialmente um processo na máquina hospedeira. O Docker implementa o isolamento de recursos através de namespaces em nível de núcleo. Ele permite a comunicação de processos entre hosts e containers, interferindo uns com os outros. Em comparação com as máquinas virtuais, o Docker tem as seguintes vantagens:

Primeiro, o Docker tem maior desempenho e eficiência do que

Segundo, o Docker tem menos camadas de abstração e não requer um Sistema Operacional (SO) adicional e suporte de hiper-visuais [15]. Graças a isto, o Docker tem melhor utilização de recursos. Tipicamente, pode haver milhares de Docker con- tainers rodando em uma única máquina que pode segurar apenas um pequeno número de máquinas virtuais. Devido ao peso leve do Docker, o tempo de partida precisa apenas de alguns segundos, muito mais rápido em comparação com vários minutos que uma máquina virtual precisa. Em terceiro lugar, o Docker pode funcionar em quase qualquer plataforma, o que faz com que o Docker tenha melhor mobilidade e escalabilidade [16]. Em
Além disso, é fácil de implantar e manter.

Devido às vantagens do Docker sobre as máquinas virtuais tradicionais, cada vez mais pesquisadores começam a usar o Docker em vez das máquinas virtuais [16], [17], [18], [19], [18], [19]. Por exemplo, Tihfon et al. [16] implementaram a infraestrutura de nuvem PaaS (Platform as a Service) multi-tarefa com Docker, e conseguiram uma rápida implementação de aplicações, otimização de aplicações e isolamento. Nguyen et al. [18] implementaram o clustering Message Passing Interface (MPI) distribuído para computação de alto desempenho através do Docker. A criação de clusters de MPI era originalmente muito demorada, mas com o Docker, eles tornaram este trabalho relativamente fácil. Julian et al. [19] otimizaram o cluster de rede em escala automática com Docker, e eles acreditam que os containers Docker podem ser usados mais amplamente em ambientes de produção maiores.

2.2 Algoritmo Florestal Clássico de Isolamento

métodos tradicionais de virtualização. Ao contrário da virtualização em camada dura das máquinas virtuais, Docker não tem
emulação de hardware, e implementa a virtualização no nível do sistema operacional [14].

Ao contrário de outros algoritmos, o algoritmo Isolation Forest (isto é, iForest [20]) não precisa definir um modelo matemático nem requer treinamento. Ele é um pouco semelhante à dicotomia. O iForest consiste de um número de árvores de isolamento (isto é, iTree) onde os nós da folha são todos dados únicos. Quanto mais cedo os dados forem isolados, mais esparsos eles estarão no conjunto de dados e, portanto, mais provável que sejam anormais.

Assumir que há N itens de dados no conjunto de dados. Os passos para a construção de um iTree são os seguintes:

Primeiro, obtemos n amostras dos dados N como as amostras de treinamento para esta árvore.

Em segundo lugar, selecionamos aleatoriamente uma característica, e selecionamos aleatoriamente um valor p dentro do intervalo de todos os valores desta característica como o nó raiz da árvore, e então realizamos uma divisão binária nas amostras. O valor da amostra que é menor que p é dividido no lado esquerdo do nó raiz, e o valor da amostra que é maior que p é dividido no lado direito do nó raiz.

Em terceiro lugar, repetimos o processo acima nos itens de dados à esquerda e à direita até atingir a condição de rescisão. Uma é que os dados em si não podem ser divididos (apenas uma amostra ou todas as amostras são iguais), e a outra é que a altura da árvore atinge o $toro_2(n)$.

Para fazer a detecção de anomalias, construímos um iForest que consiste de um número de iTrees. Suponha que o comprimento do caminho entre cada dado x e o nó raiz é $h(x)$, a média de todos os $h(x)$ é $E(h(x))$. $s(x; n)$ é o valor da anomalia dos dados x nas amostras n de um conjunto de dados. Calculamos isso da seguinte forma:

$$s(x; n) = 2^{\left(\frac{-E(h(x))}{c(n)}\right)} \quad (1)$$

$$c(n) = 2H(n-1) - (2(n-1)/n); \; H(k) = ln(k) + \xi. \quad (2)$$

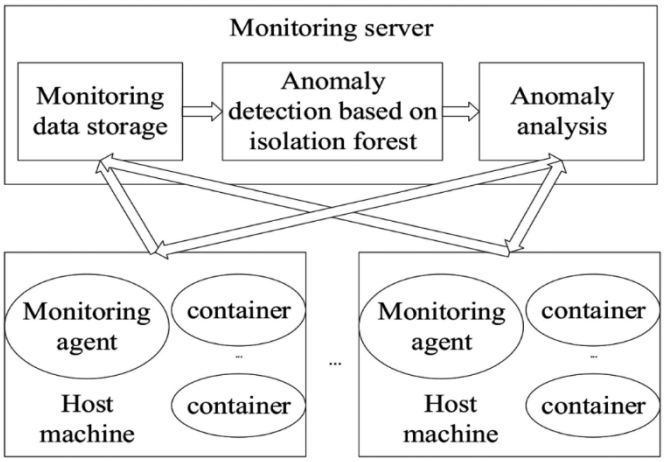


Fig. 1. Arquitetura do sistema.

O intervalo de $s(x; n)$ é [0,1]. Quanto mais próximo de 1, maior é a probabilidade de um outlier. Quanto mais próximo de 0, maior a probabilidade de x é normal. Se a maioria dos $s(x; n)$ estiver próxima a 0,5, todo o conjunto de dados é considerado como não tendo valores aberrantes óbvios.

2.3 Sistema de Monitoramento

Ganglia [6] é um projeto de monitoramento de clusters de código aberto ini- tiatizado pela UC Berkeley. O principal componente de Ganglia inclui gmond, gmetad e um front-end web. O Gmond é instalado na máquina física monitorada e é responsável pelo monitoramento da coleta de dados. Gmetad é responsável pela coleta de dados sobre os nós gmond e gmetad. O web front end pode mostrar dados em tempo real de todo o sistema de monitoramento. Entretanto, o Gmetad só pode fornecer monitoramento e não pode analisar a causa das anomalias.

Nagios [8] é um sistema de monitoramento que monitora o status operacional do sistema e as informações da rede. Ele pode monitorar hosts e serviços locais ou remotos especificados, e fornecer funções de notificação de exceção. Ele pode ser executado em uma plataforma Linux/Unix e também fornece uma interface web opcional baseada em navegador para permitir que os administradores de sistema visualizem o sta- tus da rede, vários problemas do sistema e logs. Tanto o Nagios quanto o Ganglia precisam definir um limite, o que não é adequado para monitorar um grande número de contêineres em cenas de mudanças dinâmicas.

Akshay et al. [12] propuseram um método simples de monitoração de recipientes que utiliza o próprio API do estivador para obter recursos e armazená-los no banco de dados. O método estima o desvio de stan- dard de um parâmetro de monitoramento de recursos. Os dados monitorados serão armazenados no banco de dados somente se o desvio padrão exceder um determinado limite. Este método tem mérito no armazenamento de dados, mas carece de uma função de alarme.

cAdviosr [13] é uma ferramenta de monitoramento usada pelo Google para pro-vídeo de uma função de monitoramento de recursos de múltiplos contêineres de um único nó. Como um daemon em execução, ele coleta, agrega, pro- cessam e exporta informações sobre a execução de contêineres. Ele pode obter parâmetros individuais e dados históricos de utilização de recursos para cada contêiner. Embora o cAdviosr seja fácil de configurar e possa gerar gráficos, ele só pode monitorar um host Docker e não se aplica a um ambiente de cluster de múltiplos nós. Além disso, os dados dos gráficos são apenas uma janela deslizante de um minuto. Não há função de armazenamento de dados, e não há função de alarme.

2.4 Método de Detecção de Anomalias

O método baseado na estatística matemática [9] constrói alguns modelos de distribuição padrão baseados em dados históricos, encontra pontos de dados que se desviam da distribuição, e os julga como anomalias. Entretanto, a maioria dos modelos se baseia na suposição de uma única variável. Quando a métrica de monitoramento é multidimensional, é difícil identificar com precisão a anomalia. Além disso, estes modelos são calculados usando os dados originais que contêm dados de ruído que têm um impacto significativo na construção do modelo de distribuição [21].

O método baseado na entropia da informação [22] detecta anomalias comparando as entropias do mesmo grupo em momentos diferentes. Se houver uma grande flutuação, ele indica a ocorrência de anomalias. Entretanto, este método só é adequado para um ambiente operacional estável. A mudança dinâmica do aglomerado de recipientes resultará em resultados imprecisos de detecção.

A idéia do método baseado na distância [23] é calcular a distância entre diferentes dados. Quando a distância entre dois dados é menor que uma distância D vizinha, eles são considerados como "vizinhos". Se o número de vizinhos de um dado for menor que o limite p, então os dados são julgados como dados anômalos. Entretanto, este método não é adequado para cenários em que a distribuição de dados pertence a uma estrutura multi-cluster [24]. Tipicamente, múltiplos dados anômalos contínuos de recursos métricos aparecem e se agrupam para serem vizinhos quando ocorre uma anomalia. No entanto, eles não podem ser identificados por este método.

O mais representativo dos métodos baseados na densidade é o Fator Local Outlier [10], que mede o grau de anormalidade de cada instância de dados com base no fator local outlier baseado na densidade. Quanto maior o fator local outlier, maior a probabilidade de que seja anormal. Entretanto, a estimativa da densidade local dos dados pode causar uma sobrecarga computacional significativa quando o tamanho dos dados da amostra é grande [25]. Portanto, isto não é adequado para um grande número de recipientes.

3 Projeto e Implementação do Sistema

3.1 Arquitetura

A arquitetura do sistema de monitoramento é mostrada na Fig. 1. Ela consiste principalmente de quatro componentes: *Agente de monitoramento*, *armazenamento de dados de Moni-toring*, *detecção de anomalias* e *análise de anomalias*.

Há apenas um *agente de monitoramento* em cada máquina host. Ele usa a forma não invasiva para obter a taxa de utilização de recursos do contêiner. O módulo de *armazenamento de dados de monitoramento* recebe os dados de monitoramento de cada host. Somente os dados de mono torção no período de tempo mais recente são armazenados, e os dados são organizados em um formato especificado e enviados para o módulo de detecção de anomalias. O módulo de *detecção de anomalias* detecta os dados recebidos do módulo de *armazenamento de dados de monitoramento* através de um método de avaliação de anomalias baseado no iForest-, e envia informações anormais do contêiner para o módulo de *análise de anomalias*, que primeiro obtém o registro do contêiner anormal de cada host, depois analisa o registro e localiza a causa da anomalia.

3.2 Agente de monitoramento

O projeto interno do *agente de monitoramento* é mostrado na Fig. 2. O *agente de monitoramento* coleta os dados do contêiner através do coletor de dados de monitoramento. Em seguida, o agente de monitoramento

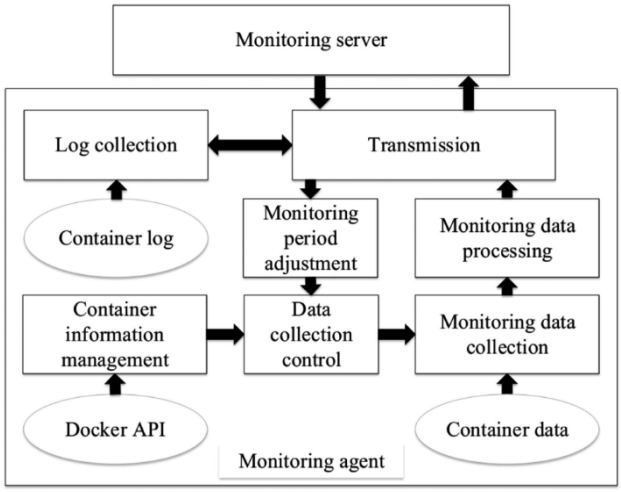


Fig. 2. Projeto interno do agente de monitoramento.

se comunica com o servidor de monitoramento, estabelece um canal de transmissão de dados e transmite os dados de monitoramento coletados através do canal. O módulo de ajuste do ciclo de monitoramento ajustará o período de monitoramento de acordo com as instruções do servidor de monitoramento.

Monitoramento da coleta de dados. Este módulo é responsável pela coleta de dados de monitoramento de todos os containers em funcionamento na máquina host. Os dados típicos de monitoramento incluem o ID do con- tainer, tempo, uso da CPU, uso da memória, velocidade de leitura/gravação do disco e velocidade da rede.

Monitoramento do processamento de dados. Este módulo recebe os dados de monitoramento com leito colado do módulo de coleta de dados de monitoramento. O módulo então realiza duas etapas. O primeiro passo é formatar os dados e encapsular a identificação do recipiente, tempo e uso de vários recursos em um formato que a base de dados pode armazenar diretamente. O segundo passo é verificar se existem recipientes espelhados de forma idêntica rodando o mesmo ser-vício e, em caso afirmativo, resumir seus dados de monitoramento.

Gerenciamento de Informações de Contêineres. Este módulo monitora principalmente as informações do estado de funcionamento do contêiner através do Docker API, incluindo a partida de novos con- tainers, o fechamento de contêineres antigos, suas identificações, informações de tarefas e informações de espelhamento. Em seguida, ele passa estas informações para o módulo de controle de coleta de dados.

Ajuste do Período de Monitoramento. O módulo mantém uma tabela de dados, que contém a identificação de cada contêiner no host e seu período de monitoramento. Ao receber o comando de ajuste do período de monitoramento enviado pelo servidor, o mod- ule muda o período de monitoramento e envia os resultados alterados para o módulo de controle de coleta de dados.

Controle de coleta de dados. Este módulo é o centro de controle do agente de monitoramento e mantém uma fila de coleta de dados. Ele calculará o próximo recipiente monitorado com base no último tempo de coleta e período de monitoramento de cada recipiente, e enviará esta informação para o módulo de coleta de dados de monitoramento. Ao mesmo tempo, o módulo também aceita as informações de partida e parada do con- tainer transmitidas pelo módulo de gerenciamento de informações do con- tainer, adicionando ou eliminando assim os contêineres na fila. O módulo também pode ajustar a seqüência de monitoramento dos containers na fila de acordo com as informações de modificação do período de monitoramento transmitidas pelo ajuste do período de monitoramento.

módulo. O período de monitoramento indica o intervalo de tempo para coletar as informações do contêiner. Quando um contêiner é considerado anormal, seu período de monitoramento é reduzido pela metade a fim de identificar a anomalia o mais rápido possível. Neste caso, as informações correspondentes ao contêiner serão coletadas com maior freqüência. Assim, o container será ajustado para uma posição na frente da fila. Em contraste, se um contêiner se recuperar ao normal, seu período de monitoramento será duplicado. O contêiner será ajustado para uma posição na parte de trás da fila.

Coleta de toras. Com base no comando de coleta de logs do servidor de monitoramento, o módulo coleta logs para o container específico e passa o log para o mod- ule de transmissão no formato especificado.

Transmissão. Ela tem principalmente duas funções: Por um lado, aceita vários comandos do servidor de monitoramento e encaminha os comandos para os módulos correspondentes. Por outro lado, ele transfere os dados de monitoramento para o servidor mono toring.

3.3 Monitoramento do armazenamento de dados

O módulo de *armazenamento de dados de monitoramento* é responsável por armazenar os dados coletados pelo *agente de monitoramento* e transmitir os dados para o módulo de *detecção de anomalias* em um formato especificado. Ele usa o InfluxDB [26] para armazenar as informações sobre o recipiente coletado. O InfluxDB é um banco de dados de tempo, eventos e métricas distribuído em código aberto. Ele suporta a transferência de dados no formato json, facilitando assim a interação de dados com o agente mono toring e o módulo de detecção de anomalias.

Uma tabela de dados é criada para armazenar todas as informações dos contêineres. Estas informações incluem o ID do recipiente, o uso da CPU, uso da memória, taxa de leitura em disco, taxa de gravação em disco, taxa de recepção de trabalho em rede, taxa de transmissão em rede do con- tiner e tempo de coleta de dados. A fim de economizar despesas de armazenamento, somente a última hora de monitoramento de dados é armazenada em o banco de dados.

O banco de dados também tem uma tabela de controle de armazenamento com três campos, o ID do container, o número de linhas na tabela de dados e o tempo da última modificação. Há três opera- tions para as informações do contêiner.

Criação e Inserção. Após receber os dados de monitoramento enviados pelo agente de monitoramento, as informações do container são inseridas na tabela de dados. Se a mesma identificação do container não for encontrada na tabela de dados, indica que os dados de monitoramento são de um container recém-aberto. O banco de dados criará uma nova linha na tabela de controle de armazenamento para adicionar a informação do novo contêiner. Se o mesmo ID for encontrado, o *número de linhas* e o tempo de modificação do recipiente correspondente na tabela de controle de armazenamento será modificado.

Eliminação. A tabela de controle de armazenamento é digitalizada a cada dez minutos. Quando se verifica que a informação de um con- tinador não foi atualizada por mais de dez minutos, considera-se que o recipiente foi fechado, e o banco de dados apaga a informação correspondente do recipiente tanto na tabela de dados quanto na tabela de controle de armazenamento.

Envio de dados para o Módulo de Detecção de Anomalias. Porque no módulo de detecção de anomalias, uma certa quantidade de dados é necessária para construir uma floresta de isolamento. Quando o valor de *num- ber de linhas* na tabela de controle de armazenamento para um recipiente atinge 100, 100 linhas de dados na tabela de dados para este recipiente são enviadas para o módulo de detecção de anomalias no formato json.

TABELA 1
Tipo de dados sujos

Categoria	Das manifestações de dados sujos
	Valor em faltaUm dos dados é nulo
	Valor de repetiçãoOs dados redundantes aparecem
Máximo ou	mínimoSuddenly os dados são muito grandes ou muito pequenos

3.4 Detecção de anomalias

3.4.1 Limpeza de dados

Devido à grande quantidade de dados do contêiner a ser coletada, pode haver perda de dados, duplicação ou mudanças no trânsito e armazenamento. Portanto, antes de construir um isolamento for- est, é necessário primeiro limpar os dados e remover os dados sujos no interior. Os tipos comuns de dados sujos são mostrados na Tabela 1:

A primeira é apagar os dados redundantes no conjunto de dados. Os dados redundantes podem afetar a estrutura das florestas de isolamento e reduzir a precisão da detecção de anomalias. Quando registros multi-preenchimento idênticos aparecem, os dados extras devem ser apagados. Além disso, a integridade do conjunto de dados deve ser pré-atendida. A ausência de dados freqüentemente ocorre em conjuntos de dados e, portanto, deve ser tratada adequadamente, caso contrário, afetará a estrutura e a precisão da detecção de anomalias das florestas de isolação. Os casos de falta grave são definidos como: a) Falta mais de 20% dos pontos de monitoramento ao longo de um período de tempo. b) Faltam 5 ou mais pontos de monitoramento consecutivos.

Se houver uma grave perda de dados no conjunto de dados, os dados nesse período são excluídos da faixa de detecção.

3.4.2 Otimização do Algoritmo Florestal de Isolamento

Introdução e Cálculo do Peso dos Recursos. A idéia do clássico algoritmo iForest tem sido muito concisa e eficiente, e pode ser aplicada diretamente a muitos cenários de aplicação. No entanto, ainda há alguns problemas quando ele é aplicado ao ambiente do contêiner. No monitoramento de contêineres, há quatro indicadores de monitoramento mais comumente utilizados: Uso de CPU, uso de memória, taxas de leitura e gravação em disco e velocidade de trabalho em rede. Quando o algoritmo iForest é aplicado ao monitoramento de contêineres, estes quatro indicadores tornam-se as medidas de fea- ture usadas para dividir o conjunto de dados. Entretanto, no clássico algoritmo iFor-est, a probabilidade de ser selecionado é a mesma para todas as características no caso aleatório. No ambiente de contêineres, as aplicações de contêineres que são intensivas em CPU são mais dependentes e sensíveis aos recursos da CPU, e as aplicações de contêineres que são intensivas em IO são mais dependentes e mais sensíveis à IO. Se os recipientes que dependem de diferentes tipos de recursos são tendenciosos para usar o mesmo padrão de monitoramento, é inevitável que a detecção de anomalias não seja imprecisa.

Portanto, este documento projeta um método de otimização. O princípio básico desta otimização é definir um valor de peso para cada um dos quatro indicadores de recursos, e depois mudar a seleção aleatória para aleatoriedade ponderada ao selecionar características na construção de árvores de isolamento. Desta forma, é mais provável que os indicadores de recursos com altos pesos sejam selecionados para a classificação dos dados do que outros indicadores - os tors. Portanto, as anomalias em recipientes que são mais dependentes e mais sensíveis a tais recursos são mais prováveis de serem encontradas.

Aqui, é proposto um método de auto-aprendizado para a otimização de recursos. Durante o uso normal de um recipiente, os parâmetros M de polarização do recipiente para cada recurso são calcu- laídos como fórmula (3):

$$M = \begin{cases} 0; & (^{Pp} N_i = 0) \\ W_0 + \dfrac{\sum_{i=1}^{pf} (N - s)}{p} & \sum_{i=1} \end{cases} . \quad (3)$$

W_0 é o valor de peso inicial da métrica do recurso, e seu valor é 1. s é o limite de recursos. N_i é a taxa de uso do recurso no momento i. p é o número de vezes a mea-certeza do uso do recurso. Se $x > 0$, então $f(x) = 1$, caso contrário $f(x) = 0$. Se o valor da métrica do recurso for sempre 0, o recipiente não utiliza o recurso. Portanto, definimos seu peso para
0. Quanto maior o parâmetro M, mais o recipiente é tendencioso em relação ao recurso.

O parâmetro de viés M é usado como o valor de peso para cada métrica de recurso. Em primeiro lugar, por padrão, todos os indicadores de recursos têm um valor de peso de 1. Em seguida, determinamos o período sob o qual o valor de peso é modificado. Especulamos - sey a cada 10 minutos como um período. O parâmetro de polarização M é calculado pela taxa de utilização de dados durante este período, e então o valor do peso é substituído por M. Finalmente, um algoritmo aleatório ponderado é usado para selecionar os valores próprios. O pseudocódigo do algoritmo é mostrado no Algoritmo 1.

Algoritmo 1. Algoritmo Aleatório Ponderado

Entrada: M_1 ; M_2 ; M_3 ; M_4 ///M_1 é peso de CPU, M_2 é peso de memória, M_3 é peso de IO, M_4 é peso de rede.
Saída: i ///A característica entre as quatro características (taxa de uso da CPU, taxa de uso da memória, taxa IO, taxa de uso da rede).
1: $M_{all} = M_1 + M_2 + M_3 + M_4$
2: $R = Random() * M_{all}$
3: para $i = 4; R > 0; i = i - 1$ do
4: $R = R - M_i$
5: fim para
6: retornar i

M_1 , M_2 , M_3 , e M_4 são os quatro valores de peso dos recursos. M_{all} é a soma de todos os valores de peso. R é um dado aleatório na faixa de 0 a M_{all} , e o último i retornado é um número de índice do recurso selecionado como recurso para dividir o conjunto de dados.

Juízo Métrico de Recursos Anomalias. O algoritmo iForest pode calcular o valor da anomalia da métrica multidimensional dos recursos, mas não pode determinar qual métrica causa a anomalia. Por exemplo, há dois tipos de casos de exceção, um é que o uso da CPU está anormalmente aumentado, e o outro é que o uso da memória está anormalmente aumentado. O valor da anomalia é semelhante em ambos os casos, utilizando o algoritmo iForest. É impossível distinguir que tipo de anomalia no uso de recursos que causou isto. A fim de resolver este problema, este artigo propõe um método para julgar a métrica da anomalia.

1) Ao construir uma árvore de isolamento, se um nó de folha é gerado quando uma divisão é realizada, o recurso selecionado pela divisão é chamado de recurso de isolamento dos dados no nó de folha, indicando que estes dados são isolados por este recurso na última divisão.
2) Definir um grupo de características de isolamento para cada dado, tais como $S(S_1 ; S_2 ;...; S_n)$. S_i representa o número de vezes

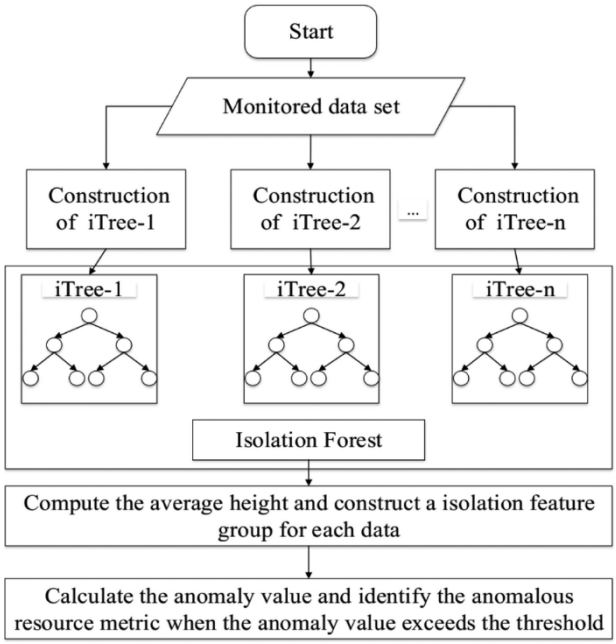


Fig. 3. Isolamento do processo de construção da floresta.

a característica métrica numerada *i* é usada como uma característica de isolamento dos dados na floresta de isolamento.

Quando construímos repetidamente árvores de isolamento e fazemos um resumo do grupo de características de isolamento para cada dado, a métrica do recurso com um valor mais alto no grupo de características de isolamento é mais provável que seja anômala do que o recurso encontrado - rico com um valor mais baixo. Assim, pode-se julgar qual métrica de recurso causou principalmente o aumento do valor da anomalia dos dados de monitoramento.

O método se baseia em uma premissa: se um valor de característica de um dado tem uma grande diferença em relação ao valor desta característica de outros dados, então, ao dividir por esta característica, é mais provável que estes dados sejam isolados separadamente. Portanto, pode-se inferir que a característica de isolamento de um dado é também a característica que tem mais probabilidade de ter o maior valor anômalo.

Quando é determinado que o recipiente é anormal, o grupo de características de isolamento dos dados anômalos de monitoramento e o grupo de características de isolamento dos dados normais de monitoramento são comparados. Calculamos a relação dos valores correspondentes das métricas nos grupos de características de isolamento. Quanto maior a relação, maior o grau de anomalia do sistema métrico.

Construção do iTree e do iForest. O iTree é uma espécie de árvore binária aleatória. Cada nódulo tem dois nódulos infantis ou é um nódulo foliar em si. Os nós foliares são dados isolados. Este artigo usa o uso da CPU do recipiente, uso de memória, taxas de leitura/escrita IO e taxa de trabalho em rede como quatro características para a construção de uma árvore de isolamento.

As etapas de construção do iTree são as seguintes:

1) Calcule o viés de cada recurso do atual con- tador com base nos dados de monitoramento, e modifique o peso da característica correspondente;
2) Selecione uma característica F entre as quatro características de recursos de contêineres. (ou seja, taxa de uso da CPU, taxa de uso da memória, taxa de IO e taxa de gravação, taxa de rede) de acordo com o Algoritmo 1;

3) Selecione aleatoriamente um valor *n* do intervalo do valor da característica F;
4) De acordo com a característica F, o conjunto de dados é dividido. Os dados com o valor da característica F menor que *n* são divididos no ramo esquerdo, e os dados com o valor da característica F maior ou igual a *n* são divididos no ramo direito.
5) Repetir os passos 2) a 4) recursivamente para construir os ramos esquerdo e direito do iTree até que as condições de dobradura sejam cumpridas:
   a) Há apenas um dado no conjunto de dados a ser dividido;
   b) A altura da árvore atinge uma altura pré-definida

Como mostrado na Fig. 3, a construção da floresta de isolamento
é um pouco semelhante à floresta aleatória. Cada parte do conjunto de dados é amostrada de forma aleatória para construir cada árvore. Então calculamos a altura média de cada dado em todas as árvores e calculamos o valor da anomalia dos dados de acordo com as fórmulas (1) e (2). Podemos ainda calcular o número de vezes que cada métrica de recurso é usada como medida de isolamento e identificar a métrica anômala do recurso.

3.4.3 Ajuste do período de monitoramento

A fim de melhorar a pontualidade do monitoramento, o período de monotorização pode ser reduzido para coletar mais dados de monitoramento para detectar mudanças no valor da anomalia dos dados de monitoramento mais cedo, no caso de possíveis anomalias. Um limiar de sensibilidade à anomalia f é definido para determinar se uma anomalia - aly é provável que ocorra. O valor de f está relacionado ao limiar d de detecção da anomalia e pode ser expresso como:

$$f = \frac{d + p}{2}. \tag{4}$$

p é o valor de anomalia normal originalmente definido para a floresta de isola- tion e é definido como 0,5 por padrão. Quando o valor médio da anomalia dos dados em um período está entre f e d, embora o critério para julgar a anomalia não seja alcançado neste momento, o alto valor da anomalia indica que o recipiente pode ser anormal. Neste momento, o container é definido como um objeto de monitoramento intensivo, e o servidor de monitoramento envia uma mensagem como {"container_id": 100 ; "tipo": inten- sive} para o agente de monitoramento. O container_id é o ID do container, e existem dois tipos: *intensivo* e *extensivo*. Quando o tipo é *intensivo*, o período de monitoramento correspondente é definido para metade do período de monitoramento inicial. Quando o valor médio do valor da anomalia dos dados for inferior a ƒ, o comando do tipo *extensivo* é enviado ao *agente de monitoramento* para ajustar o período de monitoramento ao período de monitoramento inicial.

3.5 Análise de anomalias

O módulo de *análise da anomalia* analisa principalmente o registro do recipiente anormal identificado pelo mod- ule de *detecção da anomalia*, e descobre porque a anomalia é causada. Os dados de origem para a análise da anomalia são o log coletado pelo módulo de detecção de log collection no agente de monitoramento. O módulo de *análise de anomalias* contém principalmente as duas partes a seguir.

3.5.1 Pré-processamento de logs

Antes da análise dos registros, o primeiro passo é realizar o pré-processamento dos registros. Nós extraímos apenas eventos úteis para reduzir a sobrecarga de armazenamento e análise.

TABELA 2
Informações de configuração do experimento

Configuração de	MachineHardware	Configuração de software
1	CPU Intel(R) Xeon(R) E5620 @ 2,40 GHz, 16 Núcleos, 32G RAM	Ubuntu 16.04 Docker 18.03.1-ce InfluxDB 0.13.0 MySQL 5.7 Logstash 6.2.4
2	CPU Intel(R) Xeon(R) E5620 @ 2,40 GHz, 16 Núcleos, 32 G RAM	Ubuntu 16.04 Docker 18.03.1-ce Memcached v1.5.7 CloudSuite v3.0 Logstash 6.2.4

TABELA 3
Espécies de Anomalias

Classificação das anomalias	Ilustração
Anomalias sobre CPU	Loop infinito, bloqueio de spin
Anomalias sobre memória	Vazamento de memória, estouro de memória
Anomalias sobre disco	Agendamento inadequado de disco, explosão de log
Anomalias sobre rede	Ataque de rede, congestionamento de rede

O registro do sistema do container é salvo diretamente no formato json, o que irá gerar um grande número de seqüências de fuga, como o ⊠0008. Isto aumenta muito a quantidade de logs. Portanto, o processo de filtragem correspondente deve ser realizado em tais seqüências de fuga. Há também muitos eventos no log de aplicação que não estão relacionados à análise de exceções. Por exemplo, os logs da aplicação web registram os logs de acesso (como o acesso em arquivos jpg) que não têm efeito sobre a análise de anomalias. E esta parte precisa ser filtrada. A operação específica de filtragem dos logs é configurar a correspondência de expressão regular no plugin de filtro do arquivo de configuração do log [27], e então usar a operação drop para excluir o conteúdo correspondente do log correspondente. Em seguida, os dados de log filtrados serão armazenados no banco de dados.

3.5.2Análise de Log

A principal função do módulo de análise de log é extrair os conjuntos de itens frequentes dos eventos de log pré-processados, compará-los com o banco de dados de regras, descobrir os eventos de log que causaram as exceções e atualizar o banco de dados de regras. A base de dados de regras inclui dois tipos: a base de dados de regras normal e a base de dados de regras de exceção. As regras do banco de dados de regras normal representam os conjuntos de itens frequentes gerados quando o contêiner está funcionando normalmente. As regras do banco de dados de regras de exceção são divididas em dois tipos. Um é uma regra empírica de exceção, que é uma condição de filtragem de exceção adicionada pela experiência, tal como um nível de log ERROR, ou uma expressão regu- lar que pode encontrar um evento de log anormal típico por correspondência. A outra é uma regra de anomalia histórica, que é obtida através da filtragem dos itens freqüentes do log que foram previamente analisados e causados pelo administrador.

O fluxo básico de análise dos registros é o seguinte:

Primeiro, comparamos o log armazenado no banco de

4 AVALIAÇÃO EXPERIMENTAL

4.1 Ambiente Experimental

Fazemos experimentos tanto em ambientes simulados como em nuvens reais...

ronments. Para o ambiente de nuvens simuladas, nós implantamos

servidor de monitoramento em uma máquina, e agente de monitoramento

dados com as regras empíricas de exceção no banco de dados de regras de exceção. Se a correspondência for bem sucedida, o alarme de evento de log é emitido. Por outro lado, o algoritmo Apriori [28] é usado para extrair os conjuntos de itens freqüentes na transação do log.

Em segundo lugar, combinamos os conjuntos de itens frequentemente minerados com as regras normais e as regras históricas de exceção. Se corresponderem às regras normais, são filtrados. Se corresponder às regras de exceção históricas, é emitido o alarme de evento de registro correspondente aos conjuntos de itens freqüentes.

Terceiro, se nenhuma das partidas for bem-sucedida, o adminis- trator seleciona os conjuntos de itens frequentes e os adiciona ao banco de dados de regras normais e ao banco de dados de regras de exceção.

e o recipiente Docker na outra máquina. A configuração...
A tabela 2 mostra as informações. Para o verdadeiro ambiente de nuvens, adotamos o serviço EC2 da Amazônia [29]. Utilizamos dois tipos de configurações. Um tipo é chamado de t3. médio com 2 núcleos de CPU e 4 GB de RAM. Outro tipo é gratuito e é chamado de t3.small com uso limitado de 1 núcleo de CPU e 2 GB de RAM. Ambas as plataformas rodam Ubuntu 16.04 e Docker 18.03.1-ce. Todos os componentes de monitoramento rodam na plataforma de nuvem.

Demonstramos o sistema de monitoramento com dois pontos de referência repreensivos em ambiente de nuvem. Um deles é Memcached, e o outro é Web Search in CloudSuite. Memcached é um sistema de cache de objetos de memória distribuída, de alta performance e de código aberto, destinado ao uso em aplicações web dinâmicas de aceleração, aliviando a carga do banco de dados [30]. CloudSuite é um conjunto de referência para serviços em nuvem e consiste em oito aplicações que foram selecionadas com base em sua popularidade nos data centers atuais [31]. O benchmark Web Search é um deles e se baseia na estrutura do mecanismo de busca Apache Solr. Ele contém um índice de 12 GB que foi gerado pelo rastreamento de um conjunto de sites com o Apache Nutch. Para Memcached, usamos Mutilate [32] como um gerador de carga de trabalho, e para Web Search, usamos Cli- ent Faban fornecido pela CloudSuite como um gerador de carga de trabalho.

Uma vez que não existe um padrão de referência para as anomalias de contêineres, dividimos a anomalia em quatro categorias comuns que envolvem diferentes métricas de recursos. Elas são mostradas e ilustradas na Tabela 3.

Similar ao trabalho anterior [33], usamos os quatro casos a seguir para simular as anomalias.

Loop infinito na CPU. Injetamos esta falha na aplicação inserindo código adicional para chamar a ferramenta de estresse [34], que pode simular um loop infinito na CPU e assumir uma utilização da CPU de 100%.

Vazamento de memória. O código injetado aloca a memória de pilha sem liberar objetos, o que pode gradualmente ocupar 100% da utilização da memória.

Falha de E/S do disco. Usamos FIO [35] para injetar operações extras de leitura e gravação em disco e simular falha de E/S em disco.

Congestionamento da rede. Simulamos o congestionamento da rede usando o wondershaper [36] para limitar a largura de banda da interface de rede espec-fiada.

4.2 A comparação de resultados da detecção de anomalias

Usamos a taxa de detecção e a taxa de falsos alarmes para avaliar o resultado da detecção de anomalias.

TABELA 4
A comparação de resultados da detecção de anomalias no Memcached e na busca na Web

	Anomalias	Original		iForestOptimizado iForest		LOF	
		Taxa de detecção	Taxa de alarme falso	Taxa de detecção	Taxa de alarme falso	Taxa de detecção	Taxa de alarme falso
Memcached	Circuito sem fim na CPU	38%	0%	100%	2%	100%	5.8%
	Vazamento de memória	30%	0%	98%	2%	85%	2.3%
	Falha na E/S do disco	46%	4.2%	76%	5%	54%	6.9%
	Congestionamento da rede	94%	2.1%	100%	2%	100%	2%
Busca na Web	Circuito sem fim na CPU	48%	7.7%	100%	5.7%	100%	12.3%
	Vazamento de memória	40%	4.8%	100%	7.4%	96%	14.3%
	Falha na E/S do disco	42%	5.7%	72%	12.2%	58%	23.7%
	Congestionamento da rede	74%	5.1%	84%	6.7%	80%	14.9%

$$\text{Taxa de detecção} = \frac{TP}{TP + FN} \times 100\% \quad (5)$$

$$P_{\text{false taxa de alarme}} = \frac{FP}{TP + FP} \times 100\%. \quad (6)$$

TP (true positive) indica o número de anomalias que são classificadas corretamente. FN (falso negativo) representa o número de anomalias que não são identificadas. FP (falso positivo) resume os comportamentos normais que foram julgados como anomalias.

A fim de testar o resultado da detecção do método proposto, dois outros métodos de detecção são usados como comparações. Um é o método original de detecção de anomalias baseado no iForest-based anomaly, e o outro é baseado no algoritmo do fator de anomalia local (ou seja, LOF [10]), que é o método de detecção de anomalias baseado na densidade mais representativo. Foram realizados 200 testes e cada uma das quatro anomalias típicas mencionadas acima é injetada 50 vezes.

As Tabelas 4 resumem o resultado da detecção de anomalias para diferentes métodos em Memcached e busca na Web, com respeito. Os resultados mostram que o iForest otimizado tem uma menor taxa de falsos alarmes no Memcached em comparação com a Busca na Web. Isto ocorre porque a métrica de recursos do recipiente Memcached sob a carga normal é muito estável. Quando uma anomalia - aly - ocorre, o valor da anomalia dos dados de monitoramento muda muito, portanto tem uma alta precisão de detecção. A flutuação na métrica de recursos do Web Server sob a carga normal não é pequena, e às vezes flutuações contínuas farão com que o valor da anomalia aumente além do limiar de detecção da anomalia, resultando em falsos alarmes.

O iForest otimizado tem uma melhoria significativa na taxa de detecção em comparação com o iForest original. Isto se deve ao fato de que a métrica anômala de recursos no iForest otimizado tem um grande peso e, portanto, é mais fácil de ser escolhida como a característica de isolamento para dividir o conjunto de dados. A altura média dos dados divididos usando o recurso de isolamento no iForest é muito pequeno, resultando em um grande valor anômalo. Assim, a taxa de detecção do iForest otimizado é alta.

O iForest otimizado tem um perfor- mance comparável ou melhor que o LOF. Por exemplo, sob a injeção de falha de E/S em disco, a taxa de detecção de LOF é significativamente menor do que a do iForest otimizado. É porque a taxa de leitura ou gravação de discos anômalos não é muito diferente da taxa normal de leitura ou gravação em disco, que tem uma pequena flutuação. Portanto, a densidade local dos dados monitorados tem apenas uma pequena mudança e, portanto, a taxa de detecção de LOF é baixa. Além disso, o LOF tem uma maior taxa de falsos alarmes em comparação com o iForest otimizado, especialmente na Busca na Web. Ele indica que o LOF é mais suscetível a flutuações - métricas de recursos de ant em tempo normal de execução.

Os experimentos acima assumem que os programas maliciosos injetados consomem 100% da CPU por loops infinitos. Entretanto, na prática, o usuário malicioso que tenta com- prometer o desempenho de todo o sistema pode usar programas maliciosos que não só tomam 100% da CPU, mas, por exemplo, 60% da CPU por um longo tempo. A Tabela 5 mostra os resultados de desempenho neste caso para dois tipos de ambientes de nuvem. Cloud1 e Cloud2 representam as diferentes plataformas de nuvens com múltiplos núcleos e um único núcleo, com respeito. Para a Cloud1, usamos a ferramenta de cerco [37] para simular o ataque web que consome 60-80 por cento dos recursos da CPU. Para a Cloud2, descobrimos que a ferramenta de cerco não pode aumentar a utilização da CPU em 60 por cento. Ao invés disso, executamos um programa com 500 mil vezes de loops. Para cada loop, o programa dorme por 0,1 milissegundos. O iForest otimizado executa a melhor taxa de detecção em ambos os dois ambientes de nuvem. Embora o iForest original não tenha alarmes falsos, ele não pode detectar a anomalia causada pelo programa malicioso na maior parte do tempo. Comparativamente, o iForest otimizado tem uma pequena taxa aceitável de falsos alarmes. A taxa de

TABELA 5
A comparação de resultados da detecção de anomalias quando o malicioso
Programa consome CPU Utilização que ultrapassa 60 por cento

alarme falso em

Plataformas	Categorias	Original iForest	Otimizado iForest	LOF
Nuvem1	Taxa de detecção	24%	100%	100%
	Taxa de alarme falso	0%	1.96%	4.76%
Nuvem2	Taxa de detecção	16%	100%	79%
	Taxa de alarme falso	0%	3.84%	11.23%

Nuvem1: Amazon EC2, t3.médio, 2 CPU, 4 GB de RAM; Cloud2: Amazon EC2, t3.pequeno, 1 CPU, 2 GB de RAM.

Cloud2 é maior do que na Cloud1 para o iForest otimizado.
A razão possível é que existem mais flutuações na métrica de recursos na Cloud2.

Em geral, o iForest otimizado tem melhores resultados de detecção de anomalias em comparação com outros dois métodos.

4.3 Um Caso de Detecção de Anomalia

Aqui está um exemplo mostrando como detectar uma anomalia no contêiner Memc-ached. Durante o período de funcionamento em Memcached

TABELA 6
Os pesos da métrica de recursos

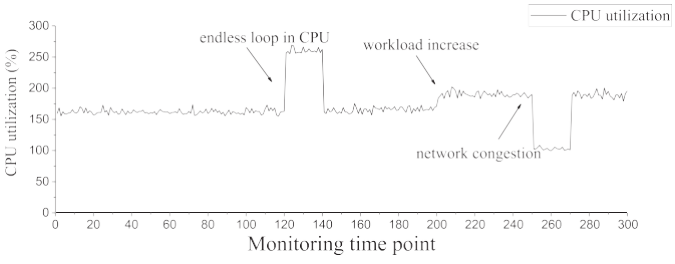
Métrica de recursos	Peso
Utilização da CPU	2
Utilização da memória	1
Taxa de leitura do disco	0
Taxa de gravação em disco	0
Taxa de recepção da rede	2
Taxa de transmissão em rede	1

três eventos são inseridos. Dois deles são anomalias, que são o loop infinito de CPU e congestionamento da rede. O outro evento é o aumento da carga de trabalho. Os pesos calculados das métricas de recursos são mostrados na Tabela 6.

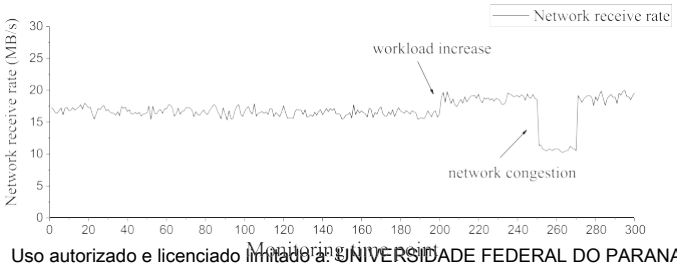
A figura 4 ilustra a utilização da CPU e a taxa de recepção da rede monitorada no tempo de execução dos containers Memcached. Observe que em um sistema com vários núcleos onde as aplicações dos recipientes estão funcionando, a utilização da CPU pode exceder 100 por cento. Na verdade, em um sistema com *n* núcleos, a utilização total da CPU do sistema pode ser de *0-n*100%* [38], [39].

A Fig. 5 ilustra a variação dos índices de anomalias calcu- lacionados de acordo com a métrica do monitor. Ela mostra que quando um loop infinito na CPU é injetado, os índices de anomalias aumentam significativamente. O valor médio dos índices de anomalias entre o ponto de tempo de monitoramento em 121 e 130 é 0,585, o que excede o limiar de detecção na linha vermelha. Assim, o recipiente é identificado como anômalo. Quando o congestionamento líquido do trabalho é injetado, os índices de anomalias aumentam significativamente. O valor médio dos índices de anomalias entre os 251 e 260 pontos de monitoramento é de 0,582, o que excede o limiar de detecção. E o contêiner é identificado como anômalo. Entretanto, o aumento da carga de trabalho não faz com que o índice de anomalias aumente e, portanto, não é identificado como uma anomalia.

A métrica anômala dos recursos precisa ser localizada após a detecção da anomalia do recipiente. Propomos um método que



(a) The CPU utilization of Memcached container at runtime



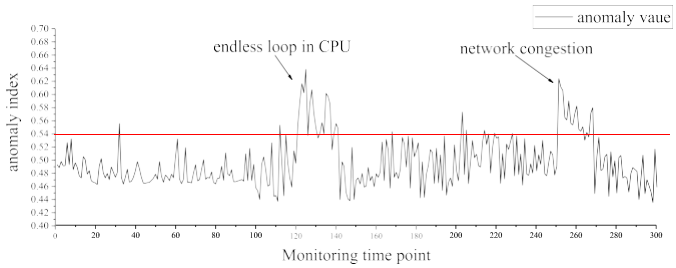


Fig. 5. Valores de anomalias do recipiente Memcached em tempo de execução. A linha vermelha mostra o limiar de detecção.

calcula a relação entre as características de isolamento na fase anômala e as características de isolamento na fase normal. A Tabela 7 mostra a relação de características de isolamento quando são injetados laços infinitos na CPU e congestionamento da rede. Pode ser visto que a razão entre as características de isolamento para características anômalas de recursos é maior do que outras. Portanto, este método pode acessar a métrica de recursos anômalos de forma rápida.

4.4 Limiar de Detecção *d*

A taxa de detecção e a taxa de falsos alarmes estão intimamente relacionadas ao limiar de detecção *d. Para* encontrar o valor ideal, foram realizados 200 testes, incluindo as quatro anomalias típicas mencionadas acima, e cada um deles foi realizado 50 vezes. Diferentes limiares de detecção foram usados para a detecção. Os resultados são mostrados na Fig. 6.

Tanto a taxa de detecção como a taxa de falsos alarmes diminuem rap- ociosamente com o aumento de *d*. Precisamos escolher o valor de *d* com uma alta taxa de detecção de anomalias e uma baixa taxa de falsos alarmes. De acordo com a Fig. 6, o valor ideal de *d* é 0,54.

4.5 O número de iTrees

O número de iTrees é um parâmetro importante no iForest opti- mized iForest. A fim de encontrar seu valor ideal, nós temos certeza da taxa de detecção e da taxa de falsos alarmes e do tempo de computação sob diferentes números de iTrees. O limiar de detecção é definido como 0,54. Os resultados são mostrados na Fig. 7.

Pode-se ver que a taxa de detecção aumenta e a taxa de falsos alarmes diminui à medida que o número de iTrees aumenta. Mas o tempo de computação ainda aumenta proporcionalmente. Aumentar o número de iTrees não melhora o efeito de detecção de anomalias depois que o número de iTrees é maior do que

100. Portanto, o valor ideal do número de iTrees é 100.

TABELA 7
Relação de Características de Isolamento quando o loop infinito na CPU e o congestionamento da rede são injetados

	Métrica de recursos	Relação de características de isolamento

(b) Network receive rate of Memcached container at runtime

Fig. 4. Métricas de recursos monitorados em tempo de funcionamento de contêineres Memcached. Observe que em um sistema de doca com *n* núcleos, a utilização total da CPU do sistema pode ser de 0-n*100% [38], [39]. O valor de *n* é 16 neste experimento.

loop infinito na CPU	Utilização da CPU	1.23
	Utilização da memória	0.81
	Taxa de recepção da rede	0.89
	Taxa de transmissão em rede	0.83
congestionamento da rede	Utilização da CPU	1.13
	Utilização da memória	0.75
	Taxa de recepção da rede	1.16
	Taxa de transmissão em rede	0.78

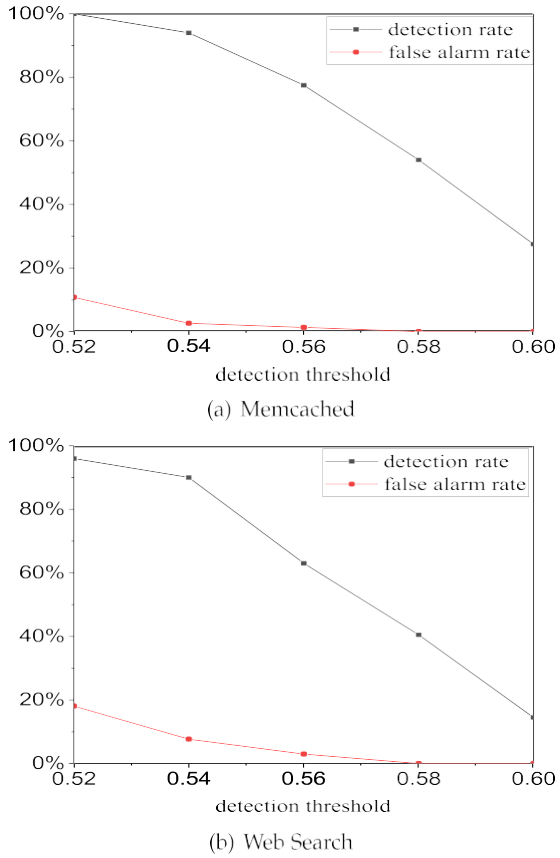
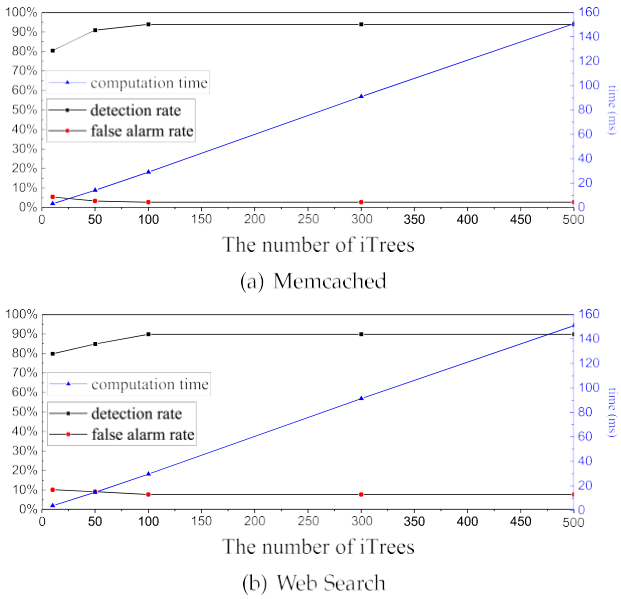


Fig. 6. Diagrama de efeito de detecção de anomalias no caso de diferentes limiares de detecção d.

4.6 Atraso de monitoramento O intervalo entre quando a anomalia é injetada e quando a anomalia é encontrada é definido como o atraso de monitoramento. Dois conjuntos de testes de detecção da anomalia baseados no iForest otimizado são realizados. Um dos testes utiliza o período de monitoramento fixo de 4 segundos, ou seja, obtemos um grupo de dados do contêiner a cada 4 segundos. O outro teste adota o método de dinamicamente



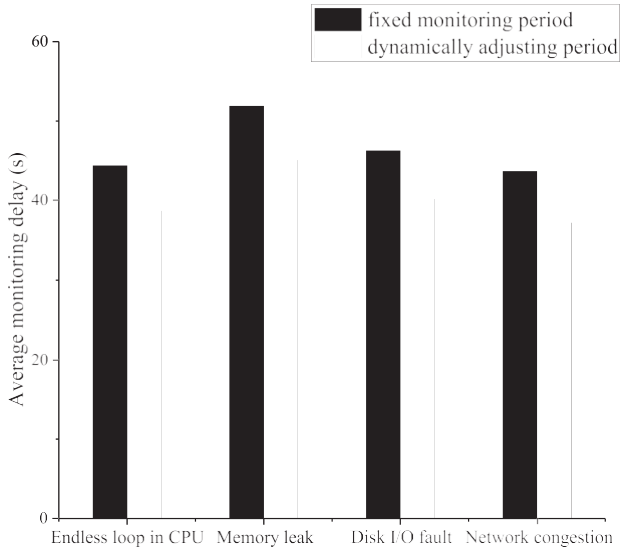


Fig. 8. Comparação da demora média de monitoramento.

Fig. 7. Resultados da detecção de anomalias com diferentes números de iTrees.

ajustando o período de monitoramento. O período inicial de monitoramento também é de 4 segundos. Injetamos quatro anomalias típicas mencionadas acima para cada teste. Os resultados da comparação são mostrados na Fig. 8.

O atraso de monitoramento do período de ajuste dinâmico é significativamente menor do que o atraso de monitoramento do período fixo de monotorização. Quando uma anomalia é identificada, o período de monitoramento é reduzido pela metade. Mais dados de monitoramento são coletados em uma unidade de tempo, fazendo com que a anomalia seja detectada mais cedo. Quando o recipiente se recupera ao estado normal, o período de monitoramento é ajustado para o valor inicial. O período de ajuste dinâmico reduz o atraso de monitoramento em uma média de
13,5 por cento.

Os atrasos médios de monitoramento estão entre 40 e 55 segundos, enquanto o ajuste do período de monitoramento é fixo em 4 segundos. O motivo é o seguinte. O algoritmo iForest otimizado obtém inicialmente 100 grupos de dados para construir um iForest. Ele tem um tamanho de janela de 100 e uma distância de deslizamento de 10. Sempre que obtém 10 novos grupos de dados, ele usa 90 grupos de dados anteriores e estes 10 novos grupos de dados para construir um novo iForest. Se o valor médio da anomalia destes 10 grupos de dados exceder o limiar de detecção, uma anomalia pode ser identificada. Como leva 4 segundos para obter um grupo de dados, é necessário um total de 40 segundos para obter estes 10 grupos de dados de contêineres. Assim, quando a anomalia destes dados é identificada, o tempo de monitoramento é de pelo menos 40 segundos. Comparativamente, quando o período de monitoramento pode ser ajustado dinamicamente, o período de monitoramento pode ser inferior a 4 segundos. Assim, o atraso de monitoramento pode ser menor do que 40 segundos às vezes.

4.7 Casos para análise de logs

Aqui estão dois exemplos mostrando como analisar os logs de contêineres. A fim de localizar a causa da anomalia através da análise das toras, duas anomalias que deixam vestígios nas toras são injetadas. Uma delas é a leitura e gravação em disco constantemente usando o carimbo do correio para simular o ataque do disco. A outra é enviar um grande número de pedidos de GET para a página web para simular o ataque da rede.

Ataque em disco. Quando o carimbo do correio está funcionando constantemente, a taxa de leitura-escrita do disco aumenta anormalmente, e o recipiente

é identificado como anômalo. Em seguida, a análise da anomalia mod- ule coleta o log do sistema de recipientes anômalos. Após o pré-processamento, o tamanho do log diminui de 476 KB para 143 KB. Em seguida, o log é armazenado no banco de dados.

O resultado da análise das regras de associação é:

Criando arquivos...Feito stdout -(freqüência*)-->>146*

Dados: stdout -(freqüência*)*-->>147

Excluindo arquivos...Feito stdout -(freqüência*)*-->>146

Ele indica que existem 146 eventos de registro, incluindo arquivos Cri- ing e 147 eventos de registro, incluindo Dados e 147 eventos de log- ging, incluindo Exclusão de arquivos. Pode ser inferido que o recipiente cria e exclui arquivos freqüentemente em fase anômala.

Ataque de rede. Nesta experiência, um recipiente nginx começa com um website rodando nele. Para simular o ataque à rede, um programa de injeção de anomalias é realizado para enviar uma massa de solicitações GET ao site. Em seguida, as taxas de envio/recepção da rede aumentam anormalmente, e o recipiente é identificado como anômalo. O módulo de análise de anomalias coleta o log de aplicação de contêineres anômalos. Após o pré-processamento, o número de eventos de registro diminui de 1434 para 723.

O resultado da análise das regras de associação é:

/ 192.168.220.1 200 GET -(freqüência*)*-->>137

Indica que a causa da anomalia é que um host cujo IP é 192.168.220.1 envia 137 pedidos de GET para o site.

5 Conclusões

Este documento propõe um sistema online de detecção de anomalias em contêineres, monitorando e analisando a métrica multidimensional dos recursos dos contêineres com base no algoritmo florestal isola- tion otimizado. Para melhorar a precisão da detecção, ele atribui a cada métrica de recurso um peso e muda a seleção do recurso run- dom no algoritmo florestal de isolamento para a seleção do recurso ponderado de acordo com o viés de recurso da aplicação do contêiner. O período de monitoramento pode ser ajustado dinamicamente de acordo com o grau de anormalidade para reduzir o atraso do monitoramento. Além disso, ele coleta e analisa as toras para a causa das anomalias. Os resultados mentais da experiência em plataformas simuladas e reais mostram que o método pode detectar com precisão anomalias no contêiner com pequenas despesas gerais de desempenho.

Agradecimentos

Agradecemos aos revisores anônimos por seus comentários construtivos. Este trabalho foi apoiado em parte pela National Science Foundation of China sob os Subsídios U1705261, 61972449, e 61821003, CCF-NSFOCUS Kun Peng research fund, Wuhan Application Basic Research Program sob o Grant No. 2017010201010104, e Hubei Natural Science and Technology Foundation sob o Grant No. 2017CFB304, e os Fundos de Pesquisa Fundamental para as Universidades Centrais Uni- sob o Grant No. 2019kfyXKJC021.

Referências

[1] A. Anwar, M. Mohamed, V. Tarasov, M. Littley, L. Rupprecht, Y. Cheng, N. Zhao, D. Skourtis, A. S. Warke, H. Ludwig, D. Hildebrand, e A. R. Butt, "Improving docker registration design based on production workload analysis", no *Proc. 16th USENIX Conf. File Storage Technol.*, 2018, pp. 265-278.

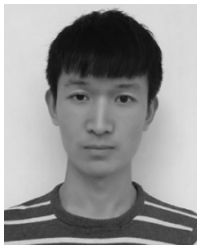
[2] "Estratégia de contêineres amazônicos examinada". [Online]. Disponível: https://www.informationweek.com/cloud/infrastructure-as-a-service/amazons-container-strategy-examined/a/d-id/1317515
[3] "Contentores IBM em Bluemix". [Online]. Disponível: https://www. ibm.com/blogs/bluemix/2015/06/ibm-containers-on-bluemix/
[4] "Munz docker occs." [Online]. Disponível: http://www.oracle.com/ technetwork/articles/cloudcomp/munz-docker-occs-3585210.html
[5] "Amazons cloud service partial failure affect certain websites". [Online]. Disponível: http://www.dailymail.co.uk/sciencetech/ article-4268850/Amazons-cloud-service-parcial-outage-affects- certain-websites.html
[6] M. L. Massie, B. N. Chun, e D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience", *Parallel Comput.*, vol. 30, no. 7, pp. 817-840, 2004.
[7] "Introdução ao Zabbix". [Online]. Disponível: http://tim.kehres. com/
[8] "Nagios": O padrão da indústria em seu monitoramento de infra-estrutura". [Online]. Disponível:https://hops://www.nagios.org/
[9] V. Chandola, A. Banerjee, e V. Kumar, "Detecção de anomalias": Uma pesquisa", *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1-58, 2009.
[10] M. M. Breunig, H. P. Kriegel, e R. T. Ng, "LOF: Identifying density-based local outliers", em *Proc. ACM SIGMOD Int. Conf. Homemidade. Data*, 2000, pp. 93-104.
[11] H. P. Kriegel, M. S. Hubert, e A. Zimek, "Angle-based outlier detection in high-dimensional data", em *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 444-452.
[12] S. A. K, J. A. K, e K. A, "Monitoramento de recursos das docas de contenção", *Int. J. Adv. Eng. Res. Develop.*, vol. 3, no. 2, pp. 146-149, 2016.
[13] "Google.cAdvisor". [Online]. Disponível: https://github.com/ google/cadvisor
[14] N. Naik, "Construir um sistema virtual de sistemas utilizando enxame de docas em múltiplas nuvens", em *Proc. IEEE Int. Symp. Syst. Eng.*, 2016, pp. 1-3.
[15] S. Mcdaniel, S. Herbein, e M. Taufer, "A two-tiered approach to I/O quality of service in docker containers", em *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, 2015, pp. 490-491.
[16] G. M. Tihfon, J. Kim e K. J. Kim, *A New Virtualized Environment for Application Deployment Based on Docker and AWS*. Cingapura: Springer, 2016.
[17] R. Liu, R. Liu, R. Liu, R. Liu, A. C. Arpaci-Dusseau, e R. H. Arpaci- Dusseau, "Slacker: Distribuição rápida com contentores de doca preguiçosos", em *Proc. Usenix Conf. File Storage Technol.*, 2016, pp. 181-195.
[18] N. Nguyen e D. Bein, "aglomerado de MPI distribuído com modo de enxame de estivadores", em *Proc. Comput. Comun. Workshop Conf.*, 2017, pp. 1-7.
[19] S. Julian, M. Shuey, e S. Cook, "Containers em pesquisa": Experiências iniciais com infra-estrutura leve", em *Proc. Xsede16 Conf. Diversity Big Data Sci. Scale*, 2016, Art. no. 25.
[20] F. T. Liu, M. T. Kai, e Z. H. Zhou, "Isolation forest", em *Proc. 8 IEEE Int. Conf. Data Mining*, 2009, pp. 413-422.
[21] N. L. D. Khoa e S. Chawla, *Robust Outlier Detection Using Commute Time and Eigenspace Embedding*. Berlim, Alemanha: Springer, 2010.
[22] M. Jiang, M. A. Munawar, T. Reidemeister, e P. A. S. Ward, "Efficient fault detection and diagnosis in complex software systems with information-theoretic monitoring", *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 4, pp. 510-522, Jul./Aug. 2011.
[23] S. Ramaswamy, R. Rastogi e K. Shim, "Efficient algorithms for mining outliers from large data sets", em *Proc. ACM SIGMOD Int. Conf. Gerenciar. Data*, 2000, pp. 427-438.
[24] F. Angiulli, S. Basta, e C. Pizzuti, "Detecção e previsão de aberrações com base na distância", *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 2, pp. 145-160, Fev. 2006.
[25] D. Pokrajac, A. Lazarevic, e L. J. Latecki, "Incremental local outlier detection for data streams", em *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2007, pp. 504-515.
[26] "Influxdb - um banco de dados de séries temporais distribuídas em código aberto". [Online]. Disponível: https://www.infoq.com/fr/presentations/ influx-db/
[27] S. Sanjappa e M. Ahmed, "Análise de logs usando logstash", no *Proc. 5ª Int. Conf. Frontiers Intell. Comput..: Teoria Aplic.*, 2017, pp. 579–585.
[28] "Algoritmo de Apriori". [Online]. Disponível:https://en.wikipedia. org/wiki/Apriori_algorithm
[29] "Nuvem de computação elástica amazônica (amazon ec2)". [Online]. Disponível em: http://aws.amazon.com/ec2
[30] "O que é Memcached". [Online]. Disponível:http://memcached. org/
[31] "Um conjunto de referência para serviços em nuvem". [Online]. Disponível: http://cloudsuite.ch/

[32] "Leverich.Mutilate". [Online]. Disponível: https://github.com/leverich/mutilar
[33] T. Wang, W. Zhang, J. Wei e H. Zhong, "Workload-aware online anomaly detection in enterprise applications with local outlier factor," in *Proc. IEEE 36th Annu. Comput. Softw. Conf. Aplic.*, 2012, pp. 25–34.
[34] "stress". [Online]. Disponível:http://people.seas.harvard.edu/apw/ stress/
[35] "FIO". [Online]. Disponível: http://freshmeat.sourceforge.net/projects/ fio/
[36] "Maravilha". [Online]. Disponível:http://lartc.org/wondershaper
[37] "Casa de cerco". [Online]. Disponível: https://www.joedog.org/siege-home/
[38] "O projeto moby". [Online]. Disponível:https://github.com/moby
[39] "Fóruns comunitários Docker". [Online]. Disponível:https://forums. docker.com/t/docker-stats-questions/811

Zhuping Zou recebeu o diploma BE em ciência da computação da Universidade Central Sul de Florestas e Tecnologia, China, em 2017 e o mestrado da Universidade Huazhong de Ciência e Tecnologia (HUST), em 2019. Suas pesquisas entre os ETs incluem o contêiner portuário e a virtualização.

Yulai Xie recebeu os diplomas de BE e PhD em ciência da computação da Huazhong University of Science and Technology (HUST), China, em 2007 e 2013, respectivamente. Ele foi bolsista visitante da Universidade da Califórnia, Santa Cruz, em 2010 e bolsista visitante da Universidade Uni chinesa de Hong Kong, em 2015. Ele é agora um professor associado da HUST, China. Seus interesses de pesquisa incluem principalmente armazenamento em nuvem e virtua- lização, proveniência digital, detecção de intrusão, aprendizado de máquinas e arquitetura de computadores. Ele é um membro do IEEE.

Kai Huang recebeu o mestrado da Universidade de Ciência e Tecnologia de Huazhong, em 2018. Entre seus interesses de pesquisa estão o docker con- tainer e a virtualização.



Gongming Xu recebeu a graduação BE em ciências com- puter do Instituto Wuhan de Technol- ogy, China, em 2018. Atualmente ele está trabalhando para o mestrado na Universidade Huazhong - Universidade de Ciência e Tecnologia (HUST).

Dan Feng recebeu os diplomas BE, ME e PhD em ciência da computação e tecnologia da Universidade de Ciência e Tecnologia de Huazhong (HUST), China, em 1991, 1994 e 1997, respectivamente. Ela é professora e diretora da divisão Data Stor- age System Division, Wuhan National Lab for Optoelectronics. Ela também é reitora da Escola de Ciência e Tecnologia da Computação (HUST). Seus interesses de pesquisa incluem arquitetura de computadores, sistemas de armazenamento massivo, sistemas de arquivos paralelos, matriz de discos e disco de estado sólido. Ela tem mais de 100 publicações em revistas e conferências internacionais, incluindo FAST, USENIX ATC, ICDCS, HPDC, SC, ICS e IPDPS. Ela é uma mem- ber do IEEE e um membro da ACM.

Darrell Long recebeu o BS em ciências com- puter pela San Diego State University, e o MS e PhD pela University of California, San Diego. Ele é um distinto professor de engenharia de computação da Universidade da Califórnia, Santa Cruz. Ele ocupa a cadeira Kumar Malavalli de Pesquisa de Sistemas de Armazenamento e é diretor do Centro de Pesquisa de Sistemas de Armazenamento. Seus interesses atuais de pesquisa incluem a área de sistemas de armazenamento, sistemas de armazenamento de alto perfor- mance, sistemas de armazenamento de arquivos e sistemas de armazenamento eficientes do ponto de vista energético. Sua pesquisa também inclui confiabilidade do sistema com- puter, vídeo-on-demand, aprendizagem aplicada de máquinas, computação móvel e segurança cibernética. Ele é membro do IEEE e membro da Associação Americana para o Progresso da Ciência (AAAS).