PyCloud:
Compartilhamento em nuvem local

Jerônimo Feijó Noble da Rosa, Msc. Eduardo M. Monks
Faculdade de tecnologia Senac Pelotas
{kastanho,emmonks}@gmail.com

***Resumo*—Com o crescimento da capacidade de armazenamento dos discos rígidos e com o seu custo caindo na mesma proporção, novos computadores já contam com HDs de grande capacidade. A ideia desse projeto é desenvolver uma solução que possa utilizar o espaço que antes estaria ocioso para realizar uma distribuição de arquivos entre diversos computadores criando uma nuvem local para aumentar disponibilidade e a redundância dos arquivos.**

I. Introdução

Nos últimos anos muito tem se falado sobre computação em nuvem e existem várias aplicações de compartilhamento de arquivos, processamento de dados e fornecimento de serviços específicos [5]. Um dos pontos positivos apresentados para o uso de computação em nuvem é a ausência de preocupação com investimento em infraestrutura, manutenção e segurança dos dados. Essas tarefas ficam a cargo do provedor de serviço contratado [3]. Para algumas organizações é um grande problema entregar a terceiros dados sensíveis, pois essas informações podem se perder, vazar ou ficarem indisponíveis por qualquer razão [2] [6].

Muitas organizações adotam outra abordagem que é adquirir a infraestrutura necessária para a própria rede local, mas nesse caso quanto maior a carga de dados mais desempenho será exigido da infraestrutura para manter essas informações disponíveis, confiáveis e seguras. Com isso é necessário à aquisição de servidores de alto desempenho que são extremamente caros e deixam a organização refém de uma tecnologia específica [1].

Se for analisada a infraestrutura existente em uma rede é possível identificar que existem PCs que são utilizados para gerar ou acessar os dados e os dispositivos de rede como servidores, roteadores, *switches* entre outros que fornecem suporte para os usuários terem disponibilidade dos serviços utilizados na organização. Considerando que a capacidade do hardware para computadores desktop muitas vezes é subutilizado e tendo em vista que o custo de aquisição é relativamente baixo. Haja visto que a capacidade de armazenamento vem crescendo nos últimos anos e que exista recurso de armazenamento não aproveitado nos equipamentos, surgiu a ideia de criar uma solução para distribuir arquivos entre computadores abstraindo do usuário para onde os dados estão sendo enviados e armazenados fornecendo alta disponibilidade e tolerância a falhas.

II. PyCloud

Esse sistema foi desenvolvido para permitir que dados sejam replicados entre diversos *hosts* na rede local aproveitando assim o espaço que em outra situação estaria ocioso. Por essa razão, o PyCloud fornece alta disponibilidade e redundância dos dados. Como cada instância do PyCloud mantém o índice completo de arquivos disponíveis, não existe um ponto único de falha. Para a recuperação dos dados bastaria apenas ter 50% dos membros ativo em redes menores (2-10 *hosts*) e no mínimo 20% em rede maiores (200 - 500 *hosts*). No decorrer do artigo será demonstrado quais procedimentos foram realizados para atingir esses valores.

Esse aplicativo foi desenvolvido em Python por isso o nome PyCloud, a tradução seria uma nuvem em python. Foi escolhida essa linguagem por ser multiplataforma, possuir muitas funções já prontas e uma documentação bem completa [4].

O PyCloud foi dividido em dois módulos, o módulo núcleo que controla todas as funcionalidades e a alteração só pode ser realizada através de um arquivo de configuração enquanto o módulo interface do usuário que é o programa que interage o usuário pode ser modificada de acordo com a necessidade do cliente. A interface do usuário se comunica com o núcleo através da comunicação entre processos e da API.

O aplicativo trabalha com comunicação *UDP* e *TCP* sendo que o primeiro utiliza três formas de operação *Multicast* ou *Broadcast* para comunicação com todos os membros e *unicast UDP* para troca individual de informação impedindo tráfego e processamento desnecessários.

O envio de arquivos é realizado através do *TCP*, devido ao comportamento do protocolo em relação a perdas e congestionamento.

Foi implementado também o versionamento de arquivos permitindo enviar arquivos com o mesmo nome, mas com conteúdos diferentes. Outra funcionalidade é a redução de desperdício impedindo que arquivos com mesmo conteúdo sejam enviados mais de uma vez.

O PyCloud fornece certo nível de segurança ao compactar, dividir em pedaços e em seguida distribuir entre diversos *hosts* onde são armazenados com um nome baseado em uma *hash*. Entrentanto, isso não impede que um usuário não autorizado possa recuperar o arquivo caso ele tenha acesso à base de dados. Nesta versão do PyCloud, não foi implementado nenhum tipo de criptografia.

III. Desenvolvimento

Nas próximas subseções serão explicados os funcionamentos dos módulos que foram propostos para esse projeto.

A. Localização de host

Diferentemente da Internet que necessita especificação de um ponto para que os *hosts* se encontrem, as redes locais permitem a localização de forma descentralizada através da utilização de mensagens *multicast* ou *broadcast*.

A função de localização no sistema PyCloud envia mensagens periódicas para informar aos outros membros da nuvem que ele está ativo, além de fornecer as informações de endereço IP, espaço disponível no disco local e outros dados.

Quando o PyCloud é iniciado esse faz um anúncio ao qual todos os membros ativos respondem com um anúncio pessoal, isso é feito com o objetivo de agilizar a convergência. Depois da primeira mensagem todas as demais não recebem resposta. Essas mensagens são utilizadas para manter o registro na tabela dos *hosts* remotos.

B. Distribuição

Este módulo exerce uma das funções mais importantes no PyCloud, pois realiza a distribuição dos pedaços de um arquivo e depois indica para qual *host* este será enviado.

Para definir a quantidade de *hosts* que devem receber um determinado pedaço utiliza-se a equação da figura 1.

$$X = 70 - \varepsilon * \theta / 100$$

Figura 1. Exemplifica como funciona a taxa de Redução

Onde Épsilon (primeira variável) é a variável de redução que permite reduzir a quantidade de *hosts* que vão receber um determinado pedaço essa variável e definida através do arquivo de configuração. Tehta (segunda variável) por sua vez representa a quantidade de *hosts* ativos no momento. Utilizando essa equação é possível garantir o crescimento da rede sem afetar a quantidade de espaço ocupado e o desempenho dela. Depois de finalizar a definição de quantos *hosts* devem receber um determinado pedaço, esses são distribuídos de forma aleatória, mas garantindo que cada membro ativo receba a quantidade de pedaços necessários.

C. Transferência de arquivos

O processo de envio é responsabilidade de várias *threads*, uma por pedaço. Cada pedaço tem a sua lista de endereços IPs que serão destinos para onde devem ser os dados devem ser enviado. Essa lista deve ser percorrida de forma sequencial e assim que um destino receba o conteúdo ele é marcado como uma origem em potencial e esse *host* já estará apto compartilhar os dados. Realizando isso permite distribuir a carga entre vários *hosts* ao invés de sobrecarregar um único. Quando é necessário baixar um arquivo, uma mensagem é enviada, todos os *hosts* que possuem algum pedaço desse arquivo respondem com o nome e número de pedaço que está armazenado localmente.

D. Controle de versões e redução de desperdício de dados

Um dos problemas que pode-se encontrar no armazenamento de dados é o gasto desnecessário de espaço com arquivos duplicados. Para solucionar esse problema foi implementado no PyCloud uma função para evitar que conteúdos duplicados sejam enviados para a nuvem. A tabela de pedaços utiliza como referência a *hash* do arquivo, permitindo assim a criação de vários registros sem que as informações sejam duplicados no sistema de arquivos.

Outra funcionalidade que pode ser utilizada é o versionamento de arquivos. Este é realizado ao se tentar o envio de um arquivo com o mesmo nome e no mesmo diretório de um já existente. O nome permanecerá o mesmo, mas novos dados serão enviados mantendo os anteriores.

IV. Testes Realizados

Nesta seção estão descritos os cenários que foram utilizado para testar as funcionalidades do PyCloud. Inicialmente, foi feita a análise do protocolo de comunicação que envolve a comunicação da aplicação. Neste teste, realizado durante 10 minutos em uma rede com um número crescente de *hosts* reais utilizando o cenário 1 da figura 2. Foi capturado o tráfego do protocolo em transmissões em *multicast* e *unicast*. Com isso foi possível precisar a quantidade de pacotes por segundo, tamanho médio de pacotes e a quantidade de dados trafegados na rede. Para

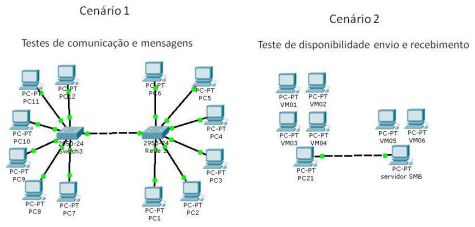

Figura 2. Mostra a estrutura utilizada nos testes.

os testes de operação do PyCloud foram utilizadas as opções de envio e recebimento de dados e depois comparados com o serviço de compartilhamento mais comuns, esses testes foram realizados com dois computadores reais utilizando maquinas virtualizadas (cenário 2 da figura 2). Foi levado em consideração fatores como disponibilidade, redundância e necessidade de espaço em disco.

V. Resultados

Essa seção inicia apresentando o resultado gerado na análise do protocolo de comunicação na tabela I.

É visível que ao se aumentar a quantidade de *hosts* nessa rede o número de pacotes trocados também crescerá, mas vale lembrar que o PyCloud foi projetado e desenvolvido para ser utilizado em redes locais e que as mensagens estarão contidas em um único domínio de *broadcast*, não sendo propagadas em outras redes. As mensagens

Tabela I
TIPOS DE MENSAGENS

hosts	Pacotes trocados	Tamanho total (bits)	Média pacotes/sec	Média bytes/sec	Tempo (sec)
3	187	14120	0,309	23,338	605
4	388	29392	0,611	46,312	634
6	600	45600	0,885	67,226	678
8	795	60420	1,175	89,263	676
12	1153	65535	1,773	134,595	650

de anúncio são relativamente pequenas com tamanho de 75 bytes, as demais também tem um tamanho semelhante exceto pela mensagem de informação de arquivo que tem 11 campos e é gerado baseado no registro de um arquivo, então não é possível precisar exatamente o seu tamanho. todas as outras mensagens *UDP* podem ser vistas na tabela II.

Tabela II
TIPOS DE MENSAGENS

Tipo de mensagem	Tamanho (bytes)	Tipo
Anúncio de *host*	75	*broadcast*
Solicitação de arquivo	87	*broadcast*
Resposta de arquivo	67	*unicast*
Alteração de registro	Dinâmico	*broadcast*

Outro teste realizado foi de verficar o tempo de convergência de uma rede utilizando o PyCloud. Esse teste foi feito em um *host* que recém havia se associado a nuvem e depois foi testado quanto tempo seria necessário para que o mesmo fosse removido da tabela de *host* e os resultado são mostrados na tabela III.

Tabela III
TABELA DE CONVERGÊNCIA

Número de *hosts*	Convergência (sec)	Remoção da tabela (sec)
3	3	73
4	3	93
6	3	68
8	3	70
12	3	72

Ao analisar as capturas de tráfego constatou-se que o *host* que já estava na rede e recebera o anúncio acrescentou o novo computador em sua tabela quase que imediatamente. No caso do *host* que acabara de entrar na rede, o tempo foi menor ou igual a 3 segundos considerando que a sua tabela estava limpa e tinha que receber a atualização dos demais. O tempo de remoção leva duas vezes e meia mais do que o tempo de anúncio (30 segundos). O resultado apresentado fica perto desse tempo esperado.

Ao capturar as mensagens de resposta dos anúncios ficou constatado que a diferença de tempo entre o recebimento da mensagem número 1 de cada *host* tem em média 0,0008 segundos. A média de tempo entre as 3 mensagens recebidas de um mesmo *host* fica em torno de 2,67 segundos, a figura 3 mostra a variação de tempo entre essas mensagem.

O resultado está dentro do esperado, pois é utilizado um tempo aleatório entre as mensagens para que um *host* não

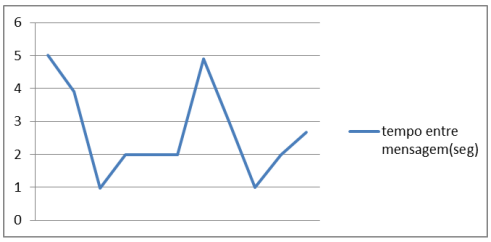

Figura 3. Tempo entre pacotes enviados por um mesmo *host*.

fique sobrecarregado com inúmeros anúncios recebidos ao mesmo tempo dos demais membros da nuvem.

Para testar o processo de distribuição foi criado um programa que adiciona o número de membros participantes e a quantidade de pedaços que deveriam ser criados. A partir do resultado é testada a disponibilidade dos dados em diferentes situações até que seja encontrado o menor valor possível sem falhas. Na tabela IV mostra o resultado dos testes para a divisão dos arquivos em 10 pedaços.

Tabela IV
TESTE DE DISPONIBILIDADE DE DADOS

Total de *Hosts*	Min. de *Hosts* necessários	% de *hosts* ativos necessários
3	2	66
10	5	50
20	7	35
50	9	18
100	10	10
500	38	7
1000	70	7

Para chegar a o resultado foram feitos 10 blocos com 100 testes cada, para que o arquivo fosse considerado recuperável todos os blocos deveriam retornar sucesso, caso contrário era aumentada a porcentagem de distribuição e testado novamente até que fosse recebido o retorno esperado.

Antes de fazer a comparação entre com as outras aplicações é necessário apresentar o atraso que é adicionado no momento anterior ao envio e posterior ao recebimento de arquivos que envolve a montagem e descompactação. A tabela V mostra esse tempo com arquivos de diversos tamanhos.

Tabela V
ATRASO ADICIONADO NO ENVIO E RECEBIMENTOS DE ARQUIVOS

Tamanho (MB)	Envio (seg)	Recebimento (seg)
3	2	1
10	7	4
50	16	7
100	72	40
150	107	55
200	146	75
750	632	432

Mesmo que seja acrescentado um tempo adicional na transmissão dos arquivos as funções que estão sendo executadas são extremamente necessárias para a operação do sistema PyCloud. Como os pedaços estão sendo arma-

zenados em ambientes não seguros a informações podem ser corrompidas ou alteradas, portanto é necessário que se utilize uma função que faça a *hash* do arquivo. Para o armazenamento local e até na transmissão é utilizada a compactação para reduzir o tráfego de rede e o espaço ocupado em disco e, por último e mais importante, a divisão de arquivos é o modo no qual o PyCloud baseia sua funcionamento.

No teste de disponibilidade e espaço ocupado, foi utilizada uma rede com o tamanho de 4 *hosts* e um arquivo de 10MB. Essa comparação utilizou serviços comuns de compartilhamento de arquivos em redes locais. Os resultados estão disponibilizados na tabela VI.

Tabela VI
TABELA DE COMPARAÇÃO ENTRE PROTOCOLOS

Protocolo	Disponibilidade	espaço ocupado/*host*
PyCloud	Varia com o tamanho da rede	7 MB
Samba	Necessidade de servidores replicados	10 MB
FTP	Necessidade de servidores replicados	10 MB
NFS	Necessidade de servidores replicados	10 MB

O teste mostrou que ao parar o serviço de compartilhamento de arquivos não foi mais possível realizar a recuperação dos dados nas aplicações testadas, como já era esperado. No sistema PyCloud os dados permaneceram disponíveis com apenas dois hosts ativos.

O último teste realizado foi para verificar em redes com tamanhos crescentes a equação de redução da figura 1 e os resultados podem ser vistos na tabela VII.

Tabela VII
TABELA DE DISTRIBUIÇÃO DE PEDAÇOS

Número de *hosts*	% pedaços/*host*	Num pedaço/*host*	Espaço ocupado (MB)/*host*
menor que 3	100%	10	10
3	70%	7	7
4	70%	7	7
5	69%	6	6
10	69%	6	6
20	68%	6	6
30	67%	6	6
50	65%	6	6
100	60%	6	6
150	55%	5	5
200	50%	5	5
500	20%	2	2
750	10%	1	1
1000	10%	1	1

É possível verificar que quanto maior a rede menor a distribuição de pedaços entre eles. Isso é feito primeiramente para não haver tráfego excessivo na rede e segundo para não ocupar uma quantidade de espaço desnecessária. Claro que isso implica diretamente na disponibilidade, mas como foi visto anteriormente na tabela de disponibilidade (tabela IV) em quase todos os caso é necessário que tenho no máximo 50% dos *hosts* ativos para recuperação dos dados.

VI. CONCLUSÕES E PROJETOS FUTUROS

O PyCloud garante a disponibilidade dos dados proporcionalmente a quantidade de hosts ativos, enquanto os outros serviços testados (Samba, FTP, NFS) necessitam de aquisição de hardware e software com o aumento de número de hosts da na rede. Para projetos futuros serão desenvolvidas outras funcionalidades, tais como capacidade de recuperação de falhas, criptografia e fazer o balanceamento e a redistribuição dos dados entre os hosts.

REFERÊNCIAS

[1] Chaves, S.(2011) A Questão dos riscos em ambientes de computação em nuvem. *Universidade de São Paulo.*

[2] Dahbur,K., Mohammad, B., Tarakji A.(2011). *A Survey of Risks, Threats and Vulnerabilities in Cloud Computing.* School of Engineering and Computing Sciences New York Institute of Technology Amman

[3] Engates J. (2012) Small businesses harness the power of cloud computing. Disponível em: <http://www.bbc.co.uk/news/business-17222816>. Acesso em: jun 2012.

[4] Goerzen, J. (2010) Fundations pf Python Network Programming. 1 Edição. Apress.

[5] Greengard, S. (2010) Cloud Computing and Developing Nations. *communications of the acm.*

[6] Weber, T. (2012) Cloud computing after Amazon and Sony: ready for primetime?. Disponível em: <http://www.bbc.co.uk/news/business-13451990>. Acesso em: jun 2012.