Benzer Ürünler

storj (storj.io)

- Kullanıcıların attıkları dosyaları peerlar arasında chunklar halinde cryptleyip bölüyorlar.
- Kullanıcılar kendi hard disk alanlarını kiralık açabiliyorlar.
- Storj kendine has para birimi ile para transferlerini gerçekleştiriyor.
- Kullanıcılar bucket denilen logical folderlar oluşturuyorlar.
- Daha sonra istedikleri fileları bu bucketlara atıp networkte dağılmasını sağlıyorlar.
- Chunklar için redundancy sağlıyorlar.
- Consensus mekanizması olarak blockchain kullanıyorlar.
- Teknoloji olarak bizle aynı, iş modeli olarak farklı.

Symform

- 170 ülkeye uzanarak petabytelar boyutunda bilgi topladılar. 5 yıl devam ettikten sonra Quantum'a(non peer to peer) satıldılar.
- Bunlarda ilk 10 GB bedava sonrasında her extra 1 GB için ya 15cent veriyorlar ya da kendi locallerinde 2GB yer açıyorlar.
- Burda şöyle bir constraint var.Kullanıcılara bilgisayarlarınızı günün %80'i açık tutmalısın diyorlar.

sia (sia.tech)

- Storj ile aynı şeyi yapmışlar. (Her konuda aynılar.Hem teknoloji hem iş modeli olarak)

MaidSafe (maidsafe.net)

- Storj ve Sia ile hemen hemen aynı şeyi yapmışlar.
- Storj ve Sia’dan farklı olarak consensus mekanizması olarak blockchain kullanmıyorlar. Close groups diye başka bir yöntem kullanıyorlar. Sistemin statetini belirlemek için tüm sistem değil, 32 nodedan oluşan bir close group bir araya geliyor. Eğer 28 node tamam derse istek valid sayılıyor.

OceanStore (oceanstore.cs.berkeley.edu)

- Bu sisteme isteyen kişiler provider olarak katılabiliyor.
- Providerların kendi aralarında bir networkü var.
- Her kullanıcı bir providera subscribe olup alan kullanabiliyor ya da satabiliyor.
- Providerlar diğer providerların networkündeki dosyaları çekip kendi subscriberları arasında dağıtabiliyor.
- Crash veya kapanma gibi durumlar için redundancy oluşturuluyor.

Google Drive ve Türevleri

- Bildiğimiz centralized cloud sistemleri.
- Güvenlik büyük sorun.(İstenildiği zaman centralized serverlarda dosyaların backuplarına ulaşılabiliyor.)
- Sınırlı alan var.(Genellikle 15GB verip gerisi kısıtlı oluyor.)

Teknolojiler

NAT Traversal

- NAT arkasındaki iki hostu konuşturmak, hostlar arasında direk bağlantı kurulamadığı için sıkıntı çıkarıyor.
- NAT arkasındaki iki hostu konuşturmak için NAT Traversal teknikleri kullanılıyor.
- **UDP ve TCP Hole Punching:** En robust çalışan yöntem bu. Merkezi bir sunucuya iki host da istek atıyor. Merkezi sunucu konuşmak isteyen sunucuların ip ve portlarını birbirine yolluyor.(Ki bu adresler address translationlarında paketlerin son çıktıkları NAT aygıtlarının ip ve portu aslında) Daha sonra uygulamalar direkt bu adresler üzerinden bağlantı gerçekleştirebiliyorlar.

Redundancy

- Redundancy Maintenance Mechanisms 3 e ayrılıyor.

    - Eager: Bir dosyanın parçalarını tutması gereken bütün peerları periodically sorgula, kayıp durumda kopyalama işlemini başlat. Redundacy sabit, bandwith usage kötü çünkü churn olanlar sebebiyle sistemden ayrılan dosyalar userlar online olduğunda sisteme tekrar girecek.
    - Deterministic Lazy: Dosyanın online parça sayısını N< #parça < M gibi bir aralıkta tutulmaya çalışılır. Amaç Churn kaynaklı overheadin minimize edilmesi. Treshold aşıldığında bütün kayıpları yenilemeye çalışmak bandwith usage da spike yaratıyor. Tresholddan sonra iş hızlı kötüleşirse toplamak zor.
    - Randomized Lazy: Parçalar random orderda sorgulanıyor. T tane online parça bulmak için T+X tane sorgu yapılıyor. Parçayı saklayan ancak ulaşılaman node sayısı(X) kadar replacement yapılıyor. Parçalar yüksek oranla online ise X küçük (az kopyalama/transfer) değilse X büyük. Üstteki ikisinin arası.

Redundancy

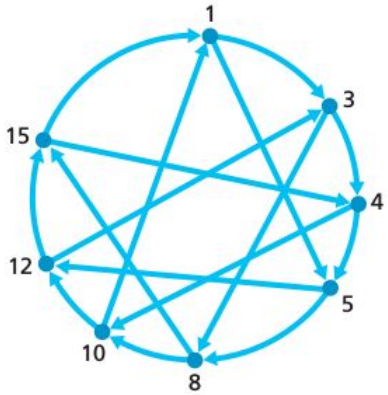
Redundancy Techniques Overview: Peerların her biri bir disk gibi düşünüldüğünde sistem RAID'i andırıyor. Ancak RAID'in yaygın versiyonları ihtiyaç duyulan gereksinimleri sağlamaktan çok uzakta. Symform RAID96 adında bir sistemle dağıtım yapmış. Bu sistem standart raid mantığına erasure coding eklenmesi. RAID96 sisteminde 64 chunktan oluşan dosya bir algoritma sonucu 96 chunka çıkarılıyor. Bu da dosyanın 32 chunka ulaşılamasa bile tekrar yapılandırılabilmesini sağlıyor. Bu redundancy bakımından naif dosya çoklamaya oranla çok daha iyi sonuç veriyor. Ancak computational overhead artıyor.

Distributed Hash Tables(DHTs)

- Normal (key,value) ikilisi tutan bir database in distributed ve P2P hali.
  - Her peer database in küçük bir kısmını tutuyor.
  - Herhangi bir peer yeni bir (key,value) ikilisi ekleyebilir.
  - Peerlar istedikleri gibi query çalıştırabilirler databasede.
  - Distributed database istenilen (key,value) ikilisini bulup aramayı yapan peer a döner.
- Bu tarz sistemlerde hem peerları hem de keyleri sayılara çevirmek peerlerın aranan datayı ve bu datayı tutan peerlerı bulmasını hızlandırıyor.
  - Örneğin oluşturulan key in kendisine en yakın peer a atılması.(key değeri 4 için 2 3 4 şeklinde numaranlandırılmış peerlardan 4 ün seçilmesi.)
- Scalability problemi için circular DHT ler kullanılıyor.

Distributed Hash Tables(DHTs)

- Scalability problemi için circular DHT ler kullanılıyor.(overlay network)
- Her peer belli bir sayıda kendisinden önce ve sonra gelen peerları tutuyor.(1 tane tutmamasının sebebi bir peerın sistemden ayrılması durumunda diğer peerlardan seçebilmesini sağlamak)
    - Bu sistemin üzerine komşu olmayan peerların arasında yeni komuşuluklar oluşturarak queryler O(logN) peerı ziyaret ederek sonuçlandırabiliyor.



File Encryption

- AES: Simetrik keyli algoritma.
- 128bit 196bit ya da 256bit'lik keyleri var.
- Keyin bit sayısı kaçsa o kadar bitlik veri üzerinde çalışıyor. (Block)
- openssl'de implement edilmiş.
- Crypto++ Library'sinde var C++ için. (https://www.cryptopp.com/)
- pycrypto library'sinde var Python için. (https://pypi.python.org/pypi/pycrypto)

File Compression

- LZO: Hıza odaklı bir algoritma. (http://www.oberhumer.com/opensource/lzo/)
- zlib: DEFLATE algoritması kullanıyor. Sıkıştırma miktarına odaklı (LZ77 + Huffman) (http://www.zlib.net/)

Algorithm	Compression Ratio	IO performance increase
LZO	41%	5%
ZLIB	48%	-16%

Line Segment Covering Problem

Given n segments of line (into the *X* axis) with coordinates [*li*;*ri*]. You are to choose the minimum number of segments that cover the segment [0;*M*]. All segment are within [0,*M*].

Can be solved greedy. Select a peer which covers time 0. After that, select peers with largest forward-extension in time.