Combining sync&share functionality with filesystem-like access

Workshop on Cloud Services for File Synchronisation and Sharing

17-18 November 2014, CERN

Kamil Guryn, Bialystok University of Technology, www.pb.edu.pl
Stanisław Jankowski, PSNC, www.psnc.pl
Maciej Brzeźniak, PSNC, www.psnc.pl

Why file-system like access + sync&share (1)

- Everyone is **excited about sync & share**:
  - It's easy to use
  - It distributes all data across user devices
  - The local sync'd folder works with no flaws with most application (even legacy ones)
  - We can have an offline access to the data assuming we managed to sync them while on-line

- **But**:
  - Research projects keep most data in the storage systems with filesystem organisation
  - Projects/ institutions have lots of data
  - Only selected data can realistically be sync'd to workstation, iPhone/Android device

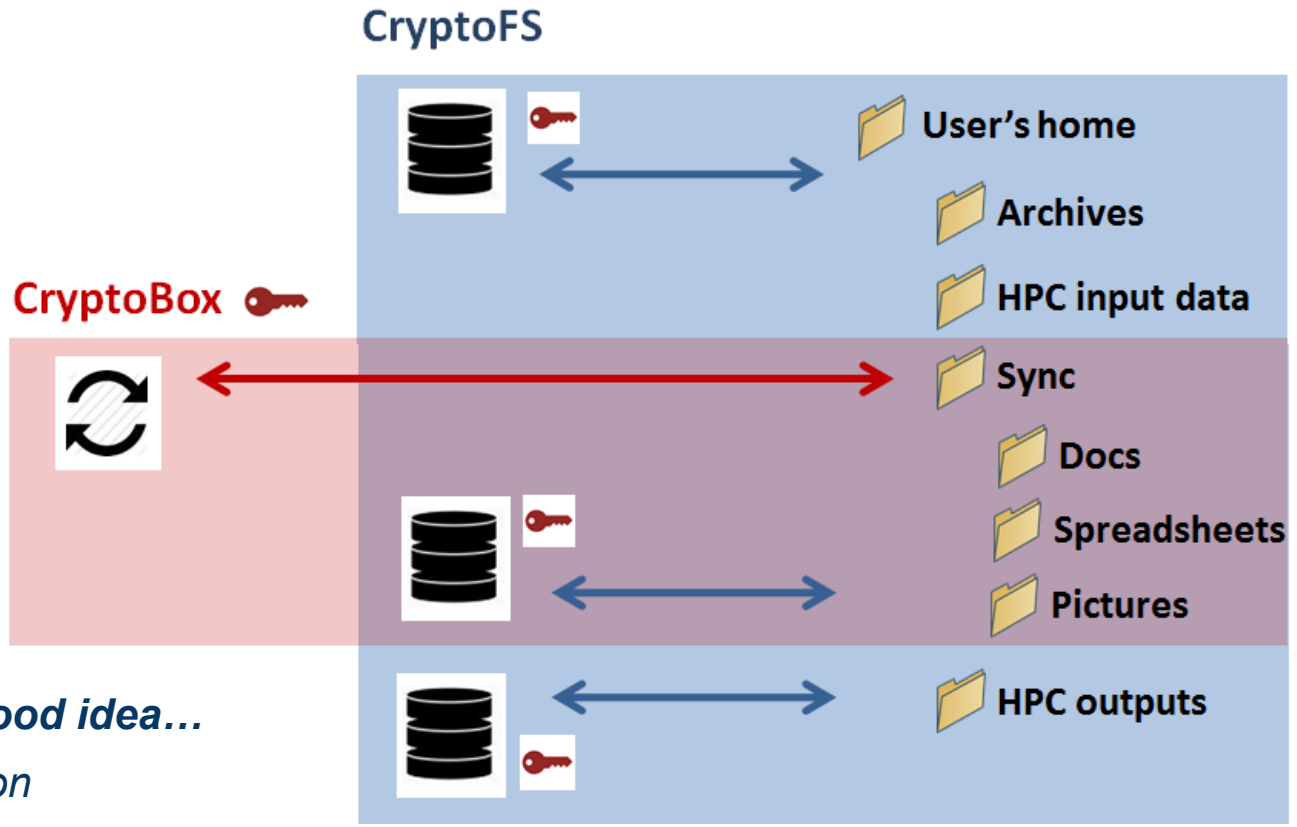
Why file-system like access + sync&share (2)

- **Other motivations to combine FS + sync&share**
  - We want to **avoid** data **redundancy hell**
    - e.g. copies of the same data for storage and sync & share systems etc.
  - We want to **re-use the same infrastructure**

    to run both sync and filesystem-like access with the ability to share
  - Projects are already doing so:
    - CERNbox is doing sync&share on top of EOS
    - dCache is running ownCloud on top of storage
    - EUDAT is implementing B2Drop based on ownCloud + FS
    - NDS2 provides CryptoBox in addition to virtual filesystems (details on the next slide)

Why file-system like access + sync&share (3)

- **NDS2 approach to FS + sync&share marriage:**
  - CryptoFS provides local filesystem-like access to the full set of remote data
  - CryptoBox syncs selected folders



Marriage is not always a good idea…

In real world no single solution is good for everything

How to combine FS access + sync&share (2)

- **Limitations/features of approaches mentioned (2)**

  - they make sense if people have / will have most of their data in file-systems, true for:
    - scientists...
    - other cases where investments were put into file systems

  - we predict that **“academics users”:**
    - university staff, researchers,
    - students

    will soon have most of their data in sync&share solutions

    because it’s handy...

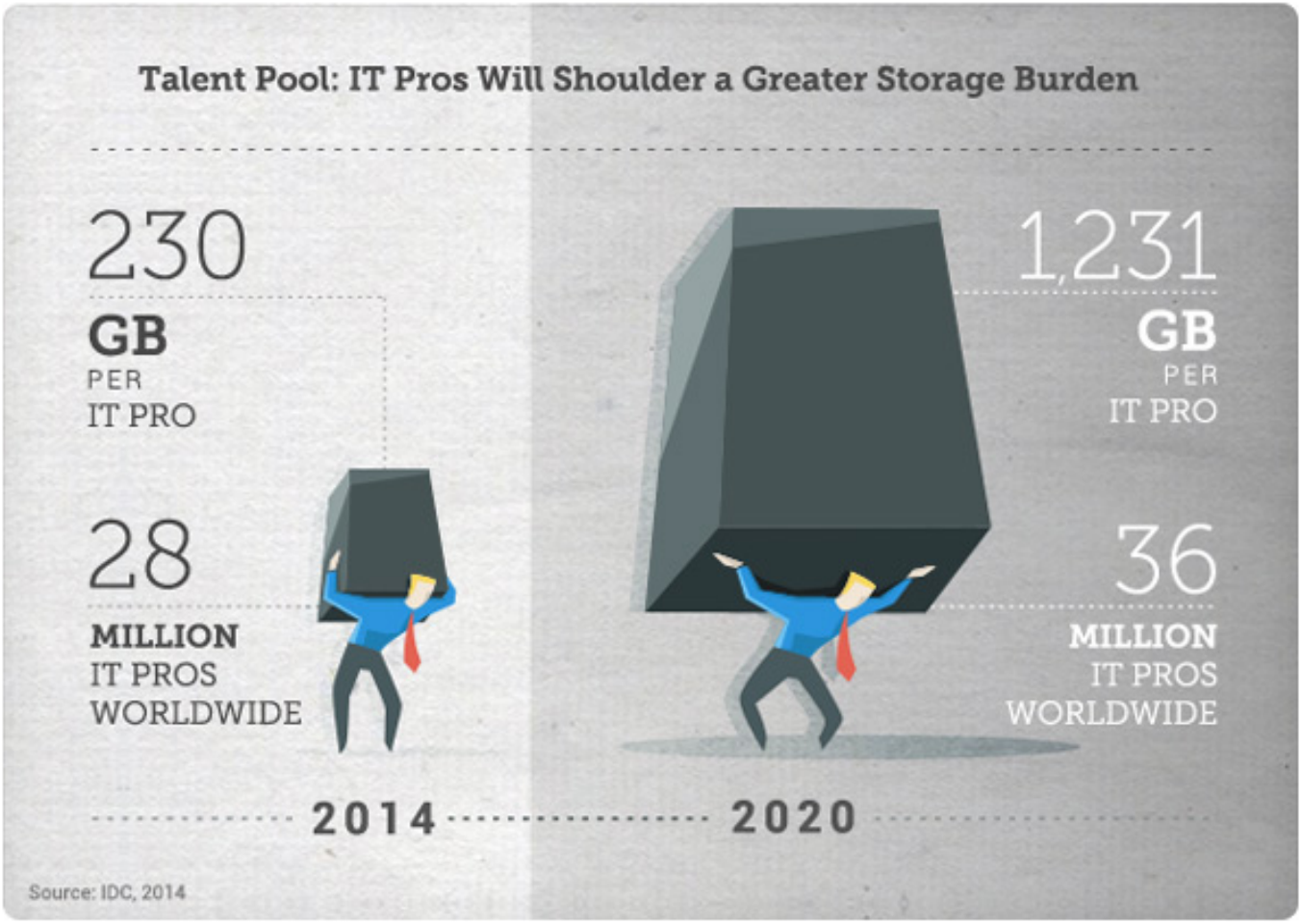
How to combine FS access + sync&share (3)

- **most data in sync & share?**

- **pros:**
  - it's feasible: easy, integrates well with user's systems, works transparently etc.
  - it's everywhere, everyone have it…

- **cons:**
  - do our today's sync&share solutions **really scale?**
  - can we:
    - efficiently and reliably deal with 40k files / 10TB of data per user?
    - run a single system country/Europe-wide? (so that people from different institutions can really share?)
    - sustain the data deluge? (see next slide)

How to combine FS access + sync&share (4)



How to combine FS access + sync&share (5)

- **Lets put things upside down...**

- **Assuming that:**
  - **there is good scalable sync&share**
  - and a lot of people 100s of 100s use it
  - have millions of files there
  - it's deployed country-wide
    (say small country ~40mln people)

- **Question:**
  - **how do I get my applications accessing large data sets?**

- **Answer:**
  - **through filesystem interface!**

How to combine FS access + sync&share (6)

- **PSNC / BIAMAN approach**

- **Take scalable sync&share e.g. Seafile:**
  - **scalable, efficient, lightweight**
  - **reliable in all scenarios**
  - git-like data organisation:
    - repositories
    - files on the client side (sync'd folder)

- **Attach filesystem interface:**
  - **Seafile...FS**
  - Extends local-like access beyond what fits into local sync'd dir
  - To some extent similar to CryptoFS, S3FS etc.

How to combine FS access + sync&share (7)

- **Why Seafile...FS is important?**
  - **local-like ad-hoc** access to remote data
    - users/applications may need files not present in local sync'd directory - typical issue of data locality
    - it is hard to predict what data will be needed in advance (if we could make it, we could sync the folder selectively)
    - calling remote data through virtual file-system may hide the fact that file is not available in sync'd folder
  - **browsing full filesystem space as local drive:**
    - no need to download the whole data set (this would be costly and slow)
    - often we don't want to sync all of the data that is shared to us by other people or company
    - yes, this is possible through WebDAV and Web Interface but...
    - some people / applications **prefer FS-like access** rather than Web(DAV) interface (e.g. legacy applications)

How to combine FS access + sync&share (8)

- **FS for Seafile vs other interfaces:**
  - **vs WebDAV:**
    - there is no free, reliable, stable clients for Windows (WebDrive paid)
    - Seafile API provides more Seafile-specific features than WebDAV, (they can be exploited by FS client + possible shell extensions)
    - **encryption:** client-side encryption would require crypto-client
    - various optimizations could be made on client side (not possible in webdav)
  - **vs Seafile's Web Interface**
    - native FS client can have similar functionality to Web Interface however packed into FS + extensions + good integration with apps
    - **encryption:** native FS client can make client-side encryption efficiently and reliably using performant encryption library (JavaScript encryption is not secure, reliable and efficient enough)
  - **vs BoxCryptor in encryption scope?**
    - **encrypted** data should be available from all clients - BoxCryptor can't make it while FS client can follow Seafile's encryption style
    - BoxCryptor would be yet another software layer on top of sync

How to combine FS access + sync&share (9)

- **Possible extensions of typical FS functionality**
  - multiple data versions presentation
  - sharing from single mount point (virtual disk)
  - client side encryption
  - various performance optimizations in client side
  - *etc.*
- **this functionality can be offered through filesystem interface (e.g. “special” dirs presenting versions)**

How to combine FS access + sync&share (10)

- **Why we are in position to make it?**
  - **Experience in implementing Virtual FS**
    - across platforms:
      - CryptoFS Windows - based on CallBackFS + Crypto layer
      - CryptoFS Linux - based on SSHFS + Crypto layer
    - reliable, production level data storage, access, sharing
    - on-the fly strong encryption and integrity control
  - **Designed and implemented secure storage & sharing logic:**
    - key hierarchy enables both **encryption** and **secure sharing**
    - examined various concepts:
      - AES-256 CTR + RSA-4096 + SHA-512
        - bulk data: 15-30 MB/s (Windows-Linux)
      - AES-256 GCM + EC:
        - bulk data: 300-400MB/s (Windows-Linux)

The roadmap

1. **Perform Seafile evaluation**
   - small-scale tests in-lab in progress
   - do large scale test planned

2. **Examine possible licensing options**
   - to be discussed with Seafile

3. **If results of 1-2 promising**
   - perform security audits/review
   - deploy Seafile on large scale in production
   - share experience with other institutions in Europe
   - exchange encryption-related ideas?
   - implement FS, test, integrate?
   - collaborate in other areas?
   - ...

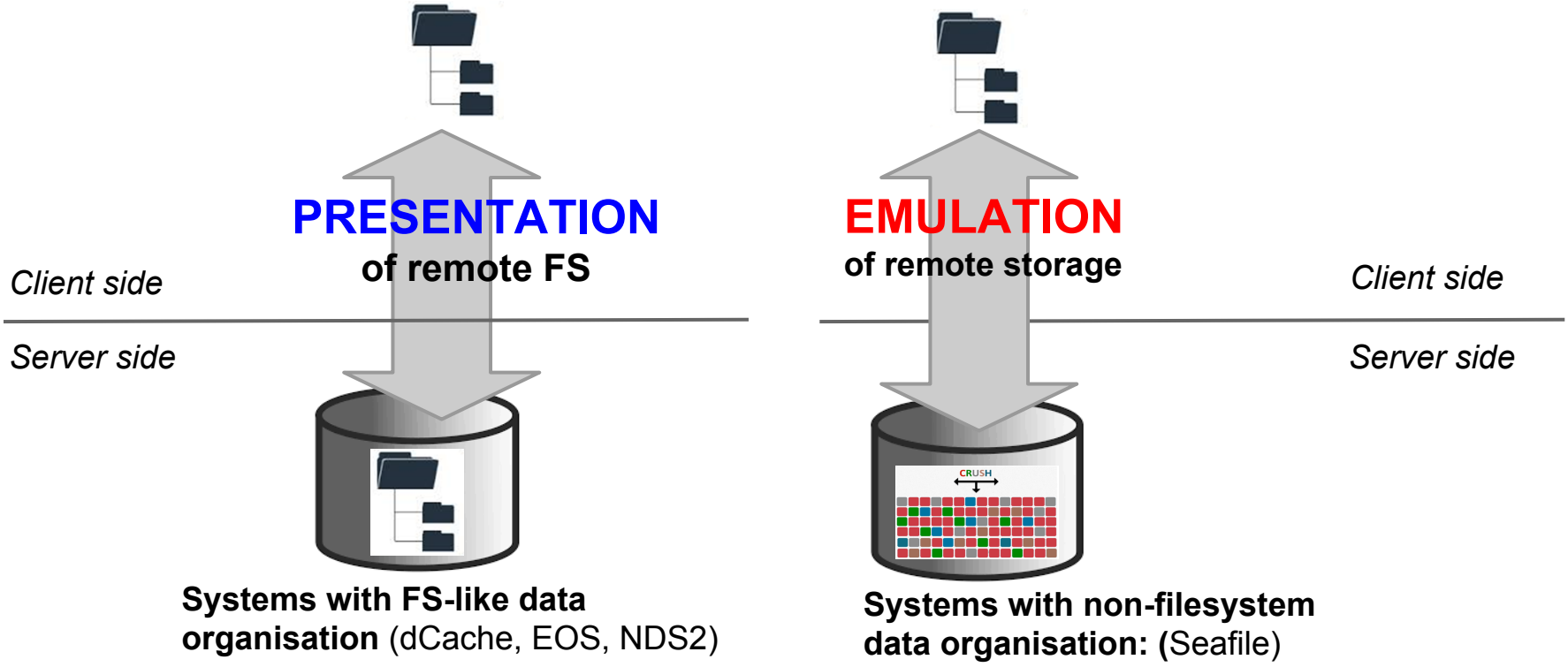
Summary (1)

At the end users want to see file-system structure:

- as they are used to filesystem-like interface

They do not care:

- data organisation on the storage/servers side
- if FS interface is implemented client- or server-side



Summary (2)

How to attack the FS + sync&share problem?

- *Marriage is not always a good solution...*
- *No solution is good for everything...*

1. **We have a filesystem logic at the application layer:**
   - then it's relatively easy to show FS on the client side
   - but we face challenges while implementing efficient and reliable sync&share solution on top of filesystem

OR

2. **We have non-filesystem logic at the application layer (repositories, indexes, commits, blocks)**
   - then we have good synchronisation
   - but we struggle implementing filesystem

We believe option 2. makes sense for future Academic Box