TOLERATING BUSINESS FAILURES IN HOSTED APPLICATIONS

Jean-Sébastien Légaré

Dutch T. Meyer, Mark Spear, Alexandru Totolici, Sara Bainbridge, Kalan MacRow, Robert Sumi, Quinlan Jung, Dennis Tjandra, David Williams-King, William Aiello, Andrew Warfield

NSS Lab. UBC
SoCC 2013
PHOTO ALBUM
You'll enjoy this
We do things better

Sign up for Lacqr

Spectaculr. Share in full resolution.
Wherevr. Available anywhere you go.

Sign In
APPS ARE GREAT. BUT THEY FAIL.

- GeoCities
- Piknik
- Google Reader
- Friendster
- And more!
EXPORT IS NOT ENOUGH

all_my_data_in_xml.tgz
You'll enjoy this FOREVER
We do things better

Sign up for Lacqr

Spectacuir. Share in full resolution.
Wherevr. Available anywhere you go.

Sign In
CONTRIBUTIONS

The Micasa platform that allows developers:

• To build apps that tolerate EOL

• To assert auditable guarantees about EOL behavior
HOSTED APPS ARE GREAT.
BUT... FAILURES ARE CATASTROPHIC

ROLLING UPGRADES
SPEED
SHARING/AC
SEARCH
VALIDATION
WE CAN BUILD DIFFERENTLY

• Cloud storage is available to *users* (Move data)
• Browsers are more powerful (Cache logic)

**ROLING UPGRADES**

**SEARCH**

**SHARING/AC**

**SPEED**

**VALIDATION**

User Data

Server data

Cloud

Browser
SHARING AND ACCESS CTL

User A's Store

A [URLs]

User B's Store

B [URL]

Nice Photo!

Append
DATA STORE DESIGN GOALS

• Allow multiple-SP ecosystem. All support API.

• Sharing: capability URLs to objects
  • No registr. to friend store.
  • Files and folders

• Revocation: undo share

• Limit writes: No RW to other stores. Append only.
  • Inbox-style communication

• Migrate between storage providers
WE CAN BUILD DIFFERENTLY

• Cloud storage is available to users (Move data)
• Browsers are more powerful (Cache logic)

ROLLING UPGRADES

SEARCH

SPEED

VALIDATION

Server Data

User Data

SHARING/AC

12
CLIENT SIDE DESIGN GOALS

• Code cached using HTML5 Application Cache

• Durable storage of client side code--can use storage provider
  • After (manual or automatic) trigger that server is gone, need to switch to "unplugged" code paths

• Library support/extension for redirecting app.com requests when unplugged
WE CAN BUILD DIFFERENTLY

• Cloud storage is available to *users* (Move data)
• Browsers are more powerful (Cache logic)

🔍 SEARCH
🔄 ROLLING UPGRADES
⏱ SPEED
👁 VALIDATION

Server Data

Search

User Data

Sharing/AC
• Users have RW/Del over their store

• Extra precautions when displaying user data
  • content integrity filters (object len, checksums)
  • routines to digitally sign and verify messages
CONTRIBUTIONS

The Micasa platform that allows developers:

• To build apps that tolerate EOL

• To assert auditable guarantees about EOL behavior
AUDITABLE PROVIDER GUARANTEES

User Store

Monitor

IDL POLICY

Server RPC

UNPLUGGED

Data RPC

3rd Party

Audit Log
EVALUATION

• Ease of development
• Satisfactory performance
• Good user experience
## APPLICATIONS BUILT

<table>
<thead>
<tr>
<th>App Name</th>
<th>SLOC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwoCans</td>
<td>1500</td>
<td>IM System</td>
</tr>
<tr>
<td>HotCRP-P</td>
<td>10K</td>
<td>Permanent HotCRP</td>
</tr>
<tr>
<td>Lenscapes</td>
<td>2200</td>
<td>Photo album sharing</td>
</tr>
<tr>
<td>Data Viewer</td>
<td>650</td>
<td>Namespace file explorer</td>
</tr>
</tbody>
</table>

Python Server prototype implementing CAPSI API (X lines of code). Supports three underlying storage backends, FS, Azure, S3.
TWOCANS

• Chat owner keeps list of message capabilities
• Message authors can revoke their messages
• Uses client-side crypto to sign and verify messages
• Very simple hosted service
  • Messages don’t go through server: p2store
  • Only user registry and public chat URLs
HOTCRP-P (PERMANENT)

- Refitted php app to DHTML view logic
- Client-side archiving of papers/reviews (copy)
- Local index is built using applet port of Apache Lucene
- Unplugged mode allows local archive search (regardless of conference website availability)
PERFORMANCE

• Application benchmark for caching, sampling Flickr pages.

• Compare loading pages statically with content-identical Micasa impl.

• Evaluation server keeps flattened capability structure.

• Compare cached vs non-cached load times and BW.

Accessing an item: $root/Comments/0/{icon, author, text}
OVERHEADS

- Fetching Micasa blobs slower than apache static fetch
- Content integrity overhead (checksum + signatures)
- Additional data dependencies
LOAD TIMES AND BW OVER STATIC

• **Page Load times:**
  • 80% of pages have <100% overhead over static (2sec vs 1sec avg)
  • With caching, all pages have <40% load times overhead

• **BW Consumption**
  • 23% overhead, 6% when cached hierarchies are available
FUTURE WORK

• Improve user data privacy
  • Confidentiality via crypto in user-defined groups
  • Monitor exfiltration of capabilities

• Ease adoption of data store API (see paper)
  • Client-side abstraction layer to support backend diversity

• Explore advertising avenues (see paper)
ALLOW TESTING GUARANTEES

• Raise level of trust between users and application providers
• Unplug to test out features present after End-of-Life (EOL)
• Provide audit mechanism
• Verify provider’s claims wrt to functionality
APPLICATION CLASSES SUPPORTED

• **Data View based Applications**: Blogs, Photo Galleries are best suited
  • Peer-to-store connections allow *sharing and commenting*

• **Local archives** of previously viewed content
  • Preserve search with client-side indexing
  • Access to “friends” data can be kept

• **Notifications** via polling (fallback for live pub-sub)

• **Server-side caching** of user objects

• **Server protocols** (e.g. SMTP for webmail)
CONCLUSION

• Platform to handle service provider EOL
• Lose no benefits from central hosting
• Application can go in unplugged mode
• App`s dependence on the provider can be audited
• Demonstrated feasibility with several useful applications
• Performance of proto well within the bounds of usability