Storing and Accessing Live Mashup Content in the Cloud

Krzysztof Ostrowski, Ken Birman

Cornell University
{krzys|ken}@cs.cornell.edu
Agenda
Agenda

1. A new, versatile storage abstraction: Checkpointed Channel (CC)

2. A new web application architecture: A web of (hyperlinked) CCs
Introduction
Storing Content in the Cloud

edge

cloud

content

server in a data center

client-server interactions

client

client

client
Storing Content in the Cloud

cloud

content

server in a data center

client

client

client

edge

client-server interactions
Storing Content in the Cloud

- **server in a data center**
- **client-server interactions**
- **content**
- **edge**
- **cloud**

**Diagram Components:**
- Clients
- Cloud
- Server
Storing Content in the Cloud

Clients connect to a server in a data center through client-server interactions. Content is stored in the cloud, accessible via a network connection at the edge of the network.
Storing Content in the Cloud

client

cloud

content

server in a data center

client-server interactions

edge
Storing Content in the Cloud
Storing Content in the Cloud

Client

Content

Server in a Data Center

Client-Server Interactions

Edge
Storing Content in the Cloud

content

update relayed through the server

server in a data center

client-server interactions

client

client

client
Storing Content in the Cloud

A server in a data center

An explicit request, e.g. HTTP GET

Clients (edge) connect to content stored in the cloud.
http://liveobjects.cs.cornell.edu
Storing Content at the Edge

- edge
- content replica
- client
- peer-to-peer interactions
- server in a data center
- content replica
- client
Storing Content at the Edge

- Server in a data center
- Edge
- Content replica
- Client
- Peer-to-peer interactions
- Content replica
- Content replica
- Content replica
Storing Content at the Edge

- edge
- content replica
- client
- peer-to-peer interactions
- content replica
- client
- server in a data center
- content replica
Storing Content at the Edge

edge

content replica

client

client

client

client

content replica

content replica

peer-to-peer interactions

content replica

server in a data center
Storing Content at the Edge

A server is located in a data center. Content is stored at the edge, with content replicas distributed among clients. Peer-to-peer interactions allow content to be updated efficiently among clients, reducing the load on the server and improving overall performance.
Storing Content at the Edge

- edge
- content replica
- updated
- multicast
- peer-to-peer interactions
- server in a data center
- content replica
- content replica
- content replica
- client
- client
Storing Content at the Edge

server in a data center

client
content replica

client
content replica

updated

updated

updated

content replica
Storing Content at the Edge

- Edge
- Content replica
- Server in a data center
- Replicated state machine
- Client

Content replica

Client
Storing Content at the Edge

- edge
- content replica
- manage membership

server in a data center
Storing Content at the Edge

server in a data center

control

data

data

clients

edge
Storing Content at the Edge

edge

server in a data center

data

control

client

client

client
Cloud vs. Edge?

Cloud content is accessed via client-server interactions with a server in a data center.

Edge content can be accessed via peer-to-peer interactions with a content replica on edge devices or through direct client-server interactions.
Cloud vs. Edge?

server in a data center

data center

Cloud vs. Edge?
edge

server in a data center

Cloud vs. Edge?

Cloud vs. Edge?

Cloud vs. Edge?

Cloud vs. Edge?
Cloud vs. Edge?

server in a data center
Cloud vs. Edge?

server in a data center

client

edge
Cloud vs. Edge?

server in a data center

edge

no replicas left
Cloud vs. Edge?

server in a data center

client-server interactions

flow of updates

bottleneck

content

cloud

edge

client

client

client
Cloud vs. Edge?

edge

cloud

content

bottleneck

server in a data center

client-server interactions

flow of updates
Cloud vs. Edge?

server in a data center

client-server interactions

content

bottleneck

flow of updates
Cloud vs. Edge?

server in a data center

client-server interactions

flow of updates

content

bottleneck
Cloud vs. Edge?

- Cloud
- Edge
- Client-server interactions
- Flow of updates
- Content
- Bottleneck
- Server in a data center
Cloud vs. Edge?

Server capacity:

• **SecondLife**: ~40 clients/server

*Second Life and the New Generation of Virtual Worlds.*
Cloud vs. Edge?

Server capacity:

- SecondLife: ~40 clients/server
Cloud ♥ Edge?

persistence and scalability
Extending Cloud to the Edge
Extending Cloud to the Edge
Extending Cloud to the Edge

- cloud
- content
- client
- server
- client-server interactions
- peer-to-peer interactions
- content
Extending Cloud to the Edge

distributed protocol (client-server, peer-to-peer, etc.)

cloud

content

server

client-server interactions

peer-to-peer interactions

content

client

client

client
Extending Cloud to the Edge
Edge is Larger than the Cloud

There are:

• 40+ times more PCs than servers (billions today, to double in five years)
• 2000+ client PCs per Google server
• 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized
Edge is Larger than the Cloud

There are:

• **40+ times more PCs than servers** (billions today, to double in five years)

• 2000+ client PCs per Google server

• 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized
Edge is Larger than the Cloud

There are:

• **40+** times more PCs than servers (billions today, to double in five years)
• **2000+** client PCs per Google server
• **40,000+** times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized
Edge is Larger than the Cloud

There are:

• 40+ times more PCs than servers (billions today, to double in five years)
• 2000+ client PCs per Google server
• 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized
There are:

- 40+ times more PCs than servers (billions today, to double in five years)
- 2000+ client PCs per Google server
- 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized
New Storage Abstraction

Unify different content access models:

- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams
New Storage Abstraction

Unify different content access models:

- **Centralized vs. replicated**
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols

- **Persistent vs. temporary**
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc

- **Server-side vs. client-side**

- **Shared public vs. shared private**
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.

- **Stateful vs. stateless**
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams
Unify different content access models:

- **Centralized vs. replicated**
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- **Persistent vs. temporary**
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- **Server-side vs. client-side**
- **Shared public vs. shared private**
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- **Stateful vs. stateless**
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams
New Storage Abstraction

Unify different content access models:
• Centralized vs. replicated
  • Centralized: web service, database
  • Replicated: distributed P2P replication protocols
• Persistent vs. temporary
  • Persistent: server in a data center
  • Temporary: collaboration session formed ad hoc
• Server-side vs. client-side
• Shared public vs. shared private
  • Public: web service or collaboration session
  • Private: session state, cookies, etc.
• Stateful vs. stateless
  • Stateful: variables, files, objects, database tables
  • Stateless: video or notification streams
New Storage Abstraction

Unify different content access models:

• Centralized vs. replicated
  • Centralized: web service, database
  • Replicated: distributed P2P replication protocols

• Persistent vs. temporary
  • Persistent: server in a data center
  • Temporary: collaboration session formed ad hoc

• Server-side vs. client-side

• Shared public vs. shared private
  • Public: web service or collaboration session
  • Private: session state, cookies, etc.

• Stateful vs. stateless
  • Stateful: variables, files, objects, database tables
  • Stateless: video or notification streams
New Storage Abstraction

Unify different content access models:

• Centralized vs. replicated
  • Centralized: web service, database
  • Replicated: distributed P2P replication protocols
• Persistent vs. temporary
  • Persistent: server in a data center
  • Temporary: collaboration session formed ad hoc
• Server-side vs. client-side
• Shared public vs. shared private
  • Public: web service or collaboration session
  • Private: session state, cookies, etc.
• **Stateful vs. stateless**
  • Stateful: variables, files, objects, database tables
  • Stateless: video or notification streams
Checkpointed Channels (CC)
Architecture

Checkpointed Channel (CC)

proxy of the channel

node₁

A₁

app. component

user

P₁

proxy of the channel

A₂

proxy of the channel

P₂

proxy of the channel

P₃

node₂

Node₃

node₃
Architecture

Checkpointed Channel (CC)

network messages

proxy of the channel

proxy of the channel

proxy of the channel

node_1

node_2

node_3

app. component

channel interfaces

user

events
Architecture

Checkpointed Channel (CC)

network messages

proxy of the channel

proxy of the channel

proxy of the channel

A1

A2

A3

P1

P2

P3

node1

node2

node3

app. component

events

channel interfaces

user
content physically resides outside the channel... 

...but it could also be cached in it

cHECKPOINTED CHANNEL

the edge may “cut through” the channel
the edge may “cut through” the channel...

content physically resides outside the channel...

...but it could also be cached in it
content physically resides outside the channel...
...but it could also be cached in it

the edge may “cut through” the channel
Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates

channel’s imported interface

channel proxy

asynchronous communication with the proxies of a checkpointed channel

submit_update(T_u update)

callpoint(T_c checkpoint)

application component

channel’s exported interface

channel<T_c, T_u>

channel<T_c, T_u>

initialize(T_c checkpoint)

update(T_u update)

request_checkpoint()
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel’s imported interface

initi aliz\(T_C\) checkpoint
update\(T_U\) update
request_checkpoint()

asynchronous communication with the proxies of a checkpointed channel

channel\(<T_C, T_U>\)
channel proxy

applicati on component

submit_update\(T_U\) update
checkpoint\(T_C\) checkpoint

channel’s exported interface
Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates

channel’s imported interface

channel’s exported interface

applicati on with the proxies of a checkpointed channel

initiate_alize(Tᶜ checkpoint)
update(Tᵘ update)
request_checkpoint()

submit_update(Tᵘ update)
checkpoint(Tᶜ checkpoint)
Interfaces

Checkpointed Communication Channel (CC)

A writable stream of checkpoints and updates

channel’s imported interface

channel’s exported interface

applicative on component

channel proxy

asynchronous communication with the proxies of a checkpointed channel

submit_update(TU update)
checkpoint(TC checkpoint)

initialize(TC checkpoint)
update(TU update)
request_checkpoint()
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel’s imported interface

channel\langle T_C, T_U \rangle

channel proxy

applicati on with the proxies of a checkpointed channel

submit_update(T_U update)
checkpoint(T_C checkpoint)

channel’s exported interface
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel’s imported interface

channel\(<T_C, T_U>\)
channel proxy

applicati on component

asynchronous communication with the proxies of a checkpointed channel

submit_update(\(T_U\ update\))
checkpoint(\(T_C\ checkpoint\))

channel’s exported interface

initi aliz(\(T_C\ checkpoint\))
update(\(T_U\ update\))
request_checkpoint()
Dynamics (local)

Checkpointed Communication Channel (CC)

initial checkpoint

channel $\langle T_C, T_U \rangle$

channel proxy

application on component

C...

C_1
Dynamics (local)

Checkpointed Communication Channel (CC)

initial checkpoint

multiple updates

channel $<T_C, T_U>$

channel proxy

applicati on on component
Dynamics (local)

Checkpointed Communication Channel (CC)

initial checkpoint

multiple updates

channel $<T_C, T_U>$

channel proxy

application on component
Interfaces

**Checkpointed Communication Channel (CC)**

a writable stream of checkpoints and updates

- channel’s imported interface
  - `initi_aliz(T_c checkpoint)`
  - `update(T_u update)`
  - `request_checkpoint()`

- application component

- channel proxy

- channel’s exported interface
  - `submit_update(T_u update)`
  - `checkpoint(T_c checkpoint)`
Dynamics (local)

Checkpointed Communication Channel (CC)
Dynamics (local)

Checkpointed Communication Channel (CC)
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel’s imported interface

channel\langle T_C, T_U \rangle

channel proxy

applicati on component

submit_update(T_U update) checkpoint(T_C checkpoint)

channel’s exported interface

asynchronous communication with the proxies of a checkpointed channel

request_checkpoints()

update(T_U update)

initialize(T_C checkpoint)
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel's imported interface

channel proxy

applicati or component

initi aliz(T_c checkpoint)
update(T_u update)
request_checkpoint()
Interfaces

Checkpointed Communication Channel (CC)
a writable stream of checkpoints and updates

channel’s imported interface

- `initi_aliz(T_C checkpoint)`
- `update(T_U update)`
- `request_checkpoint()`

asynchronous communication with the proxies of a checkpointed channel

submit_update(T_U update)
checkpoint(T_C checkpoint)

channel proxy

channel<T_C, T_U>

channel’s exported interface

applicati on component

A

C

C_1
Dynamics (local)

Checkpointed Communication Channel (CC)

request checkpoint

channel $<T_C, T_U>$

channel proxy

application on component
Dynamics (local)

Checkpointed Communication Channel (CC)

request checkpoint

channel $<T_C, T_U>$

channel proxy

application on component
Dynamics (local)

Checkpointed Communication Channel (CC)

application on component

checkpoint

channel $<T_C, T_U>$

channel proxy
Dynamics (local)

Checkpointed Communication Channel (CC)

channel $<T_C, T_U>$

channel proxy

application component

checkpoint
Dynamics (global)

Checkpointed Channel (CC)
Dynamics (global)

Checkpointed Channel (CC)

connect
Dynamics (global)

connect

Checkpointed Channel (CC)

initialize
Dynamics (global)

connect

initialize

need checkpoint
Dynamics (global)
Dynamics (global)

need checkpoint

request checkpoint
Dynamics (global)
Dynamics (global)

need checkpoint
provide checkpoint
request checkpoint
Dynamics (global)

provide checkpoint

request checkpoint

C

checkpoint

provide checkpoint

C
Dynamics (global)
Dynamics (global)

transfer checkpoint
Dynamics (global)

transfer
checkpoint

C
Dynamics (global)

checkpoint

transfer

checkpoint
Dynamics (global)

deliver checkpoint

checkpoint
Dynamics (global)
deliver
checkpoint
checkpoint
Dynamics (global)
deliver checkpoint
consume checkpoint
Dynamics (global)

consume checkpoint

wants to update
Dynamics (global)

consume checkpoint

c consumption

update

wants to update
Dynamics (global)

update

wants to update

request update
Dynamics (global)

propagate
update
Dynamics (global)
Dynamics (global)

update

propagate

update
Dynamics (global)

update

deliver update
Dynamics (global)

deliver
update
Dynamics (global)

deliver

update
Dynamics (global)

consume
update
Semantics

\[ C + U = C' \]

earlier checkpoint or state

incremental update

new checkpoint or state
Semantics

Earlier checkpoint or state + incremental update = New checkpoint or state
Semantics

\[ C + U = C' \]

- earlier checkpoint or state
- incremental update
- new checkpoint or state
Semantics

checkpoint

multiple updates

channel $\langle T_C, T_U \rangle$

channel proxy

application on component

$A \xrightarrow{C} C_1$
Semantics

$C_k + U_1 + \ldots + U_n = C_{k+1}$

channel $<T_C, T_U>$

channel proxy

application on component
Semantics
Semantics
Semantics

channel<\(T_C, T_U\)>

channel proxy

application on component
Semantics

channel\langle T_C, T_U \rangle

application on component
Semantics
Semantics

channel $\langle T_C, T_U \rangle$

application on component
Semantics

\[ C_1, C_2, C_3, \ldots, C_n, \ldots \]

"master sequence"

channel \( \langle T_C, T_U \rangle \)

application on component
submitted updates
every application observes a subset of the master sequence

Semantics

"master sequence"
every application observes a subset of the master sequence

...and nobody lags behind
Types

Checkpointed Communication Channel (CC) can be classified based on:

• the type of checkpoints,
• the type of updates,
• (and many other factors we won’t discuss)
Types

Checkpointed Communication Channel (CC) can be classified based on:

- the type of checkpoints,
- the type of updates,
- (and many other factors we won’t discuss)
Types

XML documents standardized edits on XML documents

channel\langle T_C, T_U \rangle

channel proxy

XML Channel (XC)
Types

XML documents

standardized edits on XML documents

applicati on component

channel<\(T_C, T_U\)>

channel proxy

XML Channel (XC)
Types

XML documents

standardized edits on XML documents

channel<\text{T}_C, \text{T}_U>

channel proxy

application on component

XML Channel (XC)
Types

XML documents

standardized edits on XML documents

channel $<T_C, T_U>$

channel proxy

application on component

XML Channel (XC)
Types

Checkpointed Channel (CC)

Text Channel

Binary Channel

XML Channel (XC)

RSS Channel

XHTML Channel

XAML Channel

Reference Channel

Collection<T>
http://liveobjects.cs.cornell.edu
Live Distributed Objects (LO)
Ordinary Web Applications

display

UI components
Ordinary Web Applications
Ordinary Web Applications
Ordinary Web Applications

display

UI components
Ordinary Web Applications

display

UI components
Ordinary Web Applications
Ordinary Web Applications

nested

side by side
Ordinary Web Applications

JavaFX Script

```
import javafx.ui.*;

Frame {
  title: "Bind Example 1"
  width: 300
  height: 75
  content:
    FlowPanel {
      content:
        Label {
          text: "0"
        },
        Button {
          text: "Add 1"
        },
        Button {
          text: "Subtract 1"
        }
    }
  visible: true
}
```

MXML

```
<mx:Application width="300" height="200">
  <mx:Panel title="My Application">
    <mx:Label id="myLabel"
      width="188"
      fontWeight="bold"
      fontSize="24"/>
    <mx:Button id="myButton"
      label="Click Me!"
      click="clickHandler(event);"/>
  </mx:Panel>
  <mx:Script>
    <![CDATA[
      import flash.events.MouseEvent;
      private function clickHandler(event:MouseEvent):void {
        myLabel.text = "Hello, World!";
      }
    ]]>)
  </mx:Script>
</mx:Application>
```

source: http://fxwikia.com/wiki/Introduction_to_Binding_in_JavaFX

source: http://www.adobe.com/devnet/mx/quickstart/handling_events/
Live Objects Applications

nested

front-end to back-end

side by side
Live Objects Applications

nested

front-end to back-end

data pipeline

side by side
Live Objects Applications

- nested
- front-end to back-end
- data pipeline
- side by side
- UI component
Live Objects Applications

- nested
- front-end to back-end
- side by side

Data pipeline

MPEG-2 decoder

UI component
Live Objects Applications

nested

front-end to back-end

data pipeline

side by side

MPEG-2 decoder

UI component

AES decryptor
Live Objects Applications

- AES decryptor
- MPEG-2 decoder
- UI component
- Reliable transport
- Data pipeline
- Front-end to back-end
- Nested
- Side by side
Live Objects Applications

- nested
- front-end to back-end

- data pipeline
  - unreliable transport
  - reliable transport
  - AES decryptor
  - MPEG-2 decoder
  - UI component

side by side
<Object xsi:type="ReferenceObject" id="...">
  <Parameter id="...">
    <Value xsi:type="ReferenceObject" id="...">
      <Parameter id="...">
        <Value xsi:type="ReferenceObject" id="...">
          <Parameter id="...">
            <Value xsi:type="ReferenceObject" id="...">
              ...
            </Value>
          </Parameter>
        </Value>
      </Parameter>
    </Value>
  </Parameter>
</Object>

nested

front-end to back-end

data pipeline

side by side
<Object xsi:type="ReferenceObject" id="…">
    <Parameter id="…">
        <Value xsi:type="ReferenceObject" id="…">
            <Parameter id="…">
                <Value xsi:type="ReferenceObject" id="…">
                    <Parameter id="…">
                        <Value xsi:type="ReferenceObject" id="…">
                            ...
                        </Value>
                    </Parameter>
                </Value>
            </Parameter>
        </Value>
    </Parameter>
</Object>
nested

front-end to back-end

side by side

<Object xsi:type="ReferenceObject" id="...">
  <Parameter id="...">
    <Value xsi:type="ReferenceObject" id="...">
      <Parameter id="...">
        <Value xsi:type="ReferenceObject" id="...">
          <Parameter id="...">
            <Value xsi:type="ReferenceObject" id="...">
              ...
            </Value>
          </Parameter>
        </Value>
      </Parameter>
    </Value>
  </Parameter>
</Object>
<Object xsi:type="ReferenceObject" id="...">
  <Parameter id="...">
    <Value xsi:type="ReferenceObject" id="...">
      <Parameter id="...">
        <Value xsi:type="ReferenceObject" id="...">
          <Parameter id="...">
            <Value xsi:type="ReferenceObject" id="...">
              ...
            </Value>
          </Parameter>
        </Value>
      </Parameter>
    </Value>
  </Parameter>
</Object>
<Object xsi:type="ReferenceObject" id="...">
  <Parameter id="...">
    <Value xsi:type="ReferenceObject" id="...">
      <Parameter id="...">
        <Value xsi:type="ReferenceObject" id="...">
          <Parameter id="...">
            <Value xsi:type="ReferenceObject" id="...">
              <Parameter id="...">
                <Value xsi:type="ReferenceObject" id="...">
                  <Parameter id="...">
                    <Value xsi:type="ReferenceObject" id="...">
                      ... 
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
                        ... 
                      </Value>
                      <Value xsi:type="ReferenceObject" id="...">
<Object xsi:type="ReferenceObject" id="...">
  <Parameter id="...">
    <Value xsi:type="ReferenceObject" id="...">
      <Parameter id="...">
        <Value xsi:type="ReferenceObject" id="...">
          <Parameter id="...">
            <Value xsi:type="ReferenceObject" id="...">
              <Parameter id="...">
                <Value xsi:type="ReferenceObject" id="...">
                  ...
                </Value>
              </Parameter>
            </Value>
          </Parameter>
        </Value>
      </Parameter>
    </Value>
  </Parameter>
</Object>
Live Objects Applications

- web service proxy
- membership client
- recovery protocol
- TCP transport
- UDP transport
- hole-punching
- IP multicast
- reliable multicast
- AES decryptor
- MPEG-2 decoder
- UI component
Live Objects Applications

- web service proxy
- membership client
- recovery protocol
- TCP transport
- UDP transport
- hole-punching
- IP multicast
- reliable multicast
- AES decryptor
- MPEG-2 decoder
- UI component
Live Objects Applications

- web service proxy
- membership client
- recovery protocol
- TCP transport
- UDP transport
- hole-punching
- IP multicast
- reliable multicast
- AES decryptor
- MPEG-2 decoder
- UI component
Live Objects Applications

- web service proxy
- membership client
- recovery protocol
- TCP transport
- UDP transport
- hole-punching
- IP multicast
- reliable multicast
- AES decryptor
- MPEG-2 decoder
- UI component
Live Objects Applications

data "pipeline"
Live Objects Applications

data pipeline

data pipeline
Live Objects Applications

XML
References

channel reference

activate

XML

channel proxy

P₁

P₂

P₃
References

XML points to XML points to XML points to XML points to XML.
References

XML points to

points to

XML

points to
Ordinary Web Applications

nested

XML markup

side by side
Live Objects Applications
Live Objects Applications
Live Objects Applications
Live Objects Applications
Live Objects Applications

space

plane 1

plane 2

XML
Live Objects Applications

- Space
  - Plane 1
    - Building 1
  - Plane 2
  - Building 1
- XML

Diagram with objects and connections: spaces, planes, and buildings.
Live Objects Applications
Live Objects Applications
Live Objects Applications

This is a shared text message.

Other text.

Image #1

Text 1

X: -116.12  (-263.06,77.08)  Y:  31.60  (-28.26,227.52)
Live Objects Applications

This is a shared text message.

Other text.

Text 1

Text 2

Image #1

desktop
Live Objects Applications

Other text.
Live Objects Applications

This is a shared text message.

Other text.

Text 2

Image #1

X: -116.12 (-263.06,77.08)  Y:  31.60 (-28.26,227.52)
Live Objects Applications

This is a shared text message.

Other text.

As yet another image.

Image #1

Text 1

Text 2

Desktop

Desktop 2

Image 1

Desktop

Text 2

Image 1
Live Objects Applications

This is a shared text message.

Other text.

Text 1

This is yet another message.

Text 2

Image #1

Desktop

Text 2

Image 1

Desktop 2

Text 3
Live Objects Applications

desktop 1

P1

P2

P3
Live Objects Applications
Live Objects Applications
Live Objects Applications
Live Objects Applications
Conclusions
Conclusions

1. Cloud and edge technologies can work together...
   ...but we need a new storage abstraction.

2. Checkpointed Channels (CCs)
   a) Support a variety of protocols: client-server, P2P, distributed replication, append log files, etc.,
   b) Can be used at any level of the protocol stack, at the fronted and at the backend,
   c) Can efficiently store structured mashup content,
   d) Can be hyperlinked into webs of CCs; these could serve as a basis for a new Web architecture.
Thanks