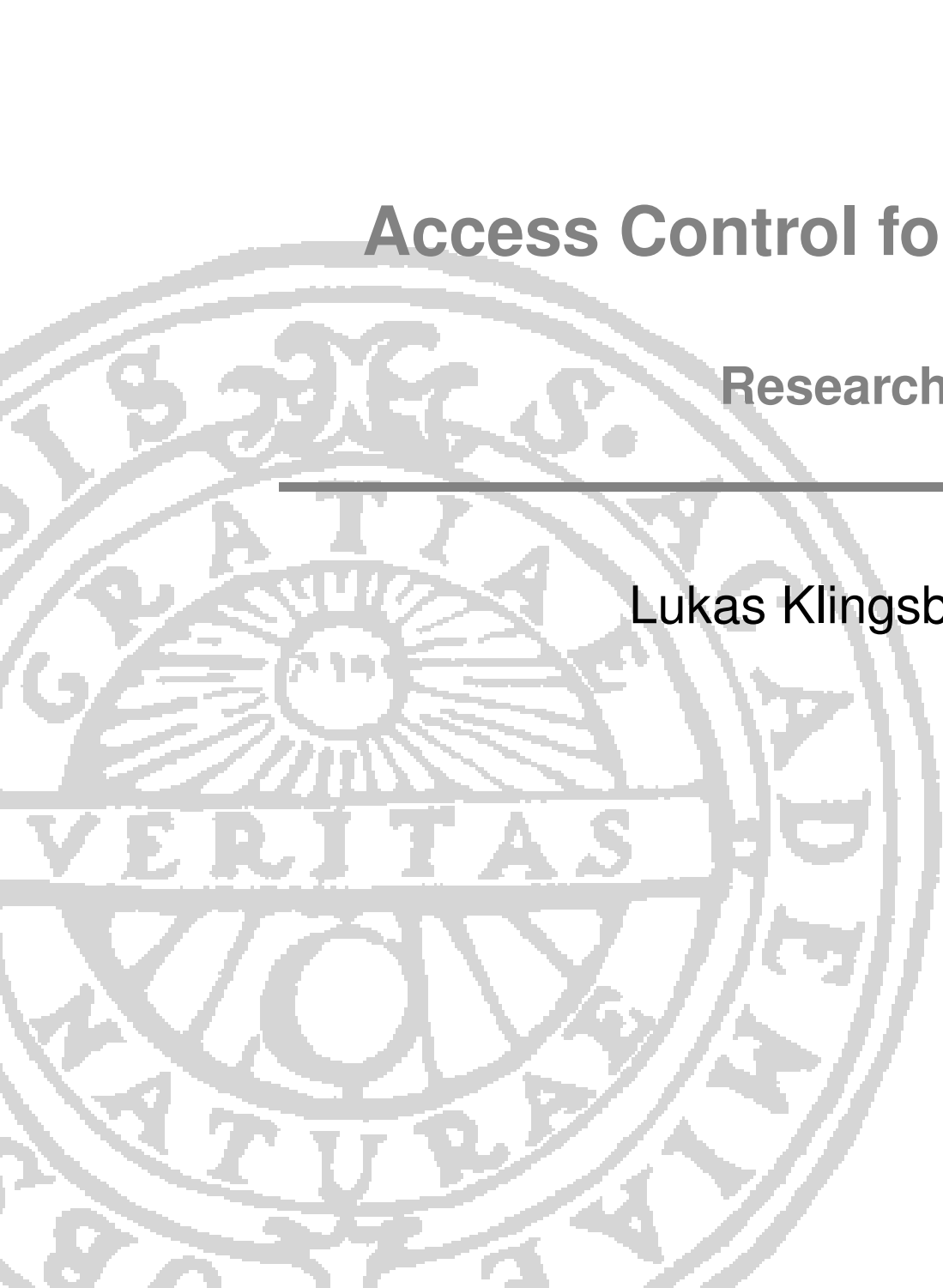


Access Control for Content Distribution Network Assets

Researching Copy-on-Write solutions

Lukas Klingsbo <lukl8671@student.uu.se>

Department of Information Technology

Uppsala University

April 20, 2016

UPPSALA
UNIVERSITET

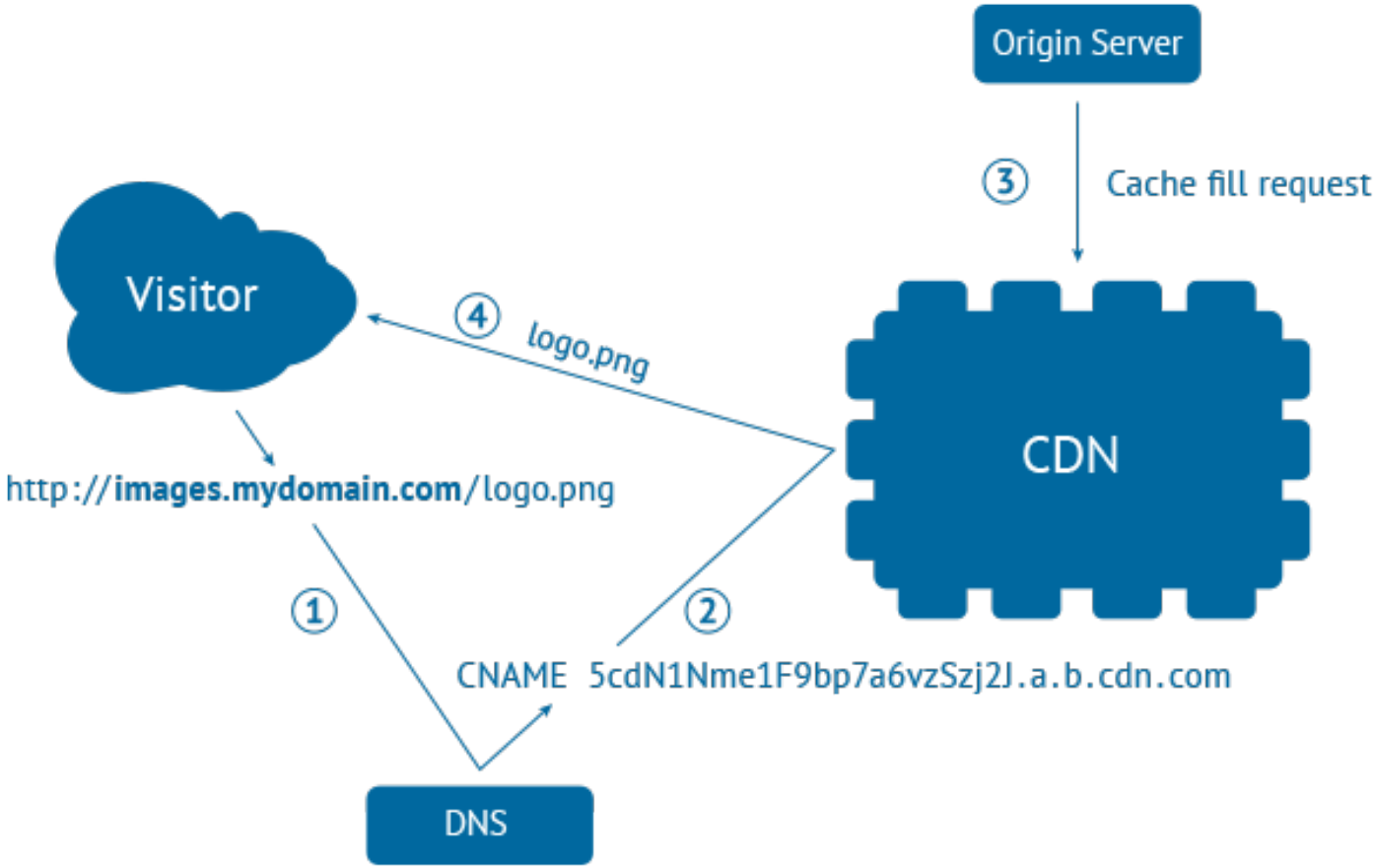
Outline

1. **Background**
   - Uprise
   - CDN
   - BattleBinary
   - Copy-on-Write
   - Related Work
2. **Problem**
   - Private Content
   - Solutions Idea
3. **Perius**
   - Model
   - Implementation

UPRISE

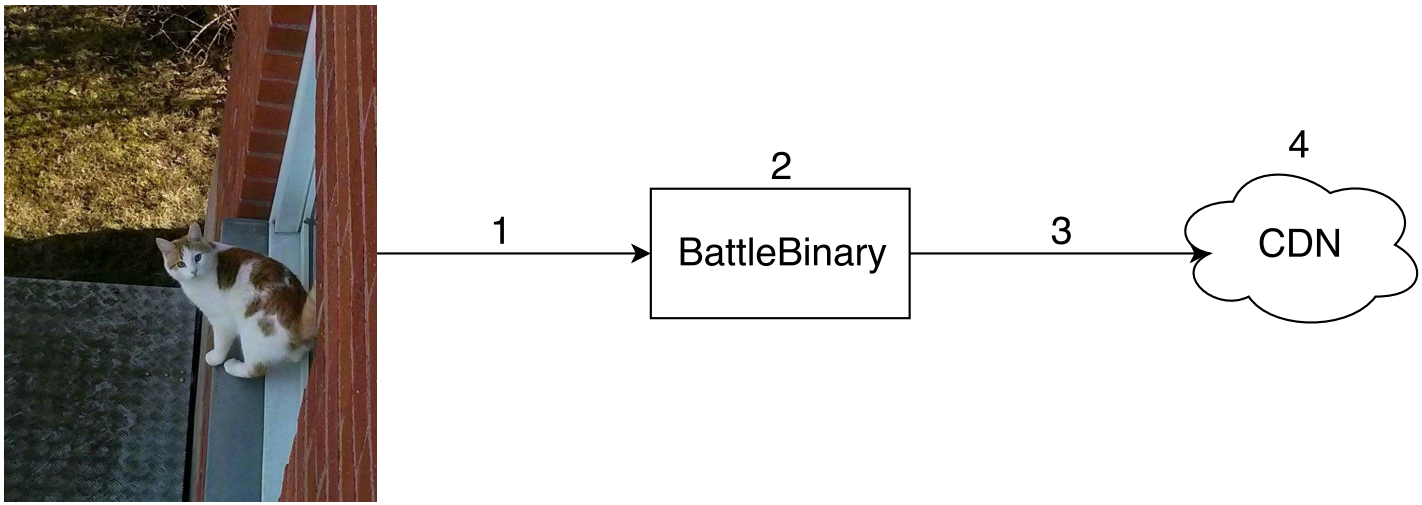
GHOST

Content Distribution Network (CDN)



BattleBinary

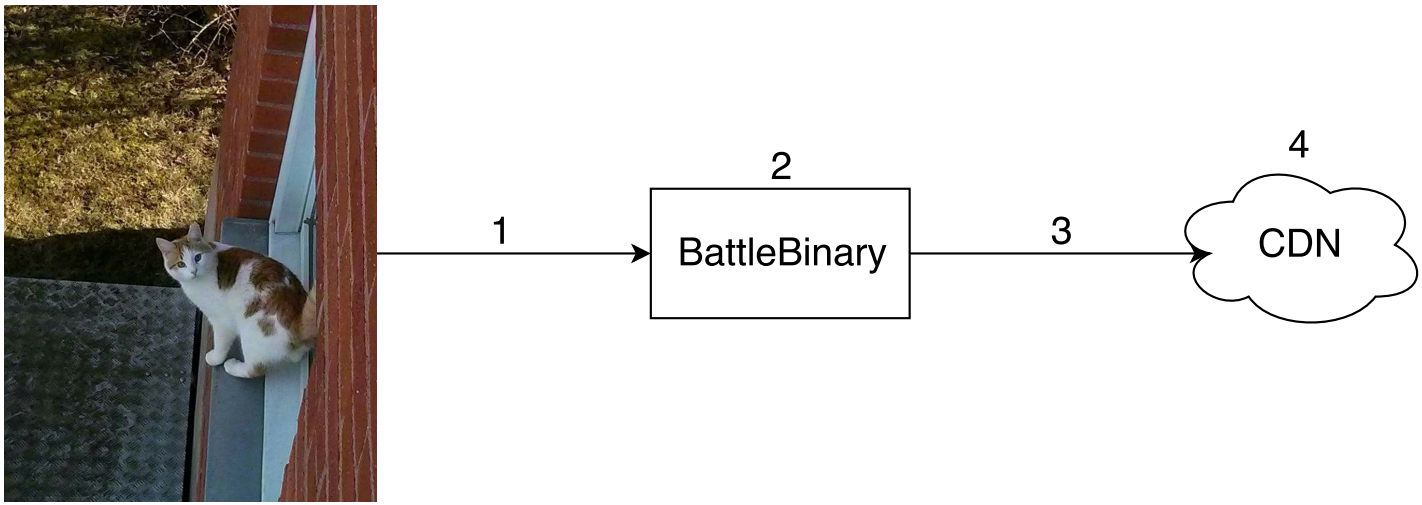
Virtual file system to organise CDN assets



- 1. Image is uploaded to BattleBinary
- 2. Filename + Part of file's hash = Asset Identifier
- 3. Upload image to CDN
- 4. Image is available to everybody with link `http://ea.akamaihd.net/cat-f1ee0283b6accd6.jpg`

BattleBinary

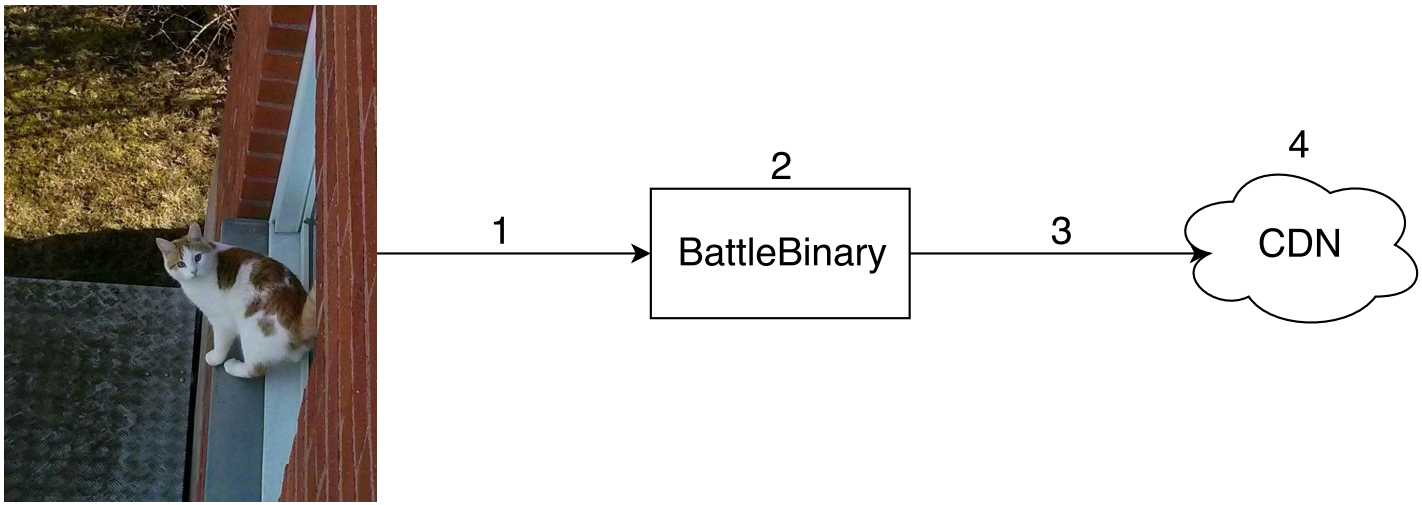
Virtual file system to organise CDN assets



- 1. Image is uploaded to BattleBinary
- 2. Filename + Part of file's hash = Asset Identifier
- 3. Upload image to CDN
- 4. Image is available to everybody with link `http://ea.akamaihd.net/cat-f1ee0283b6accd6.jpg`

BattleBinary

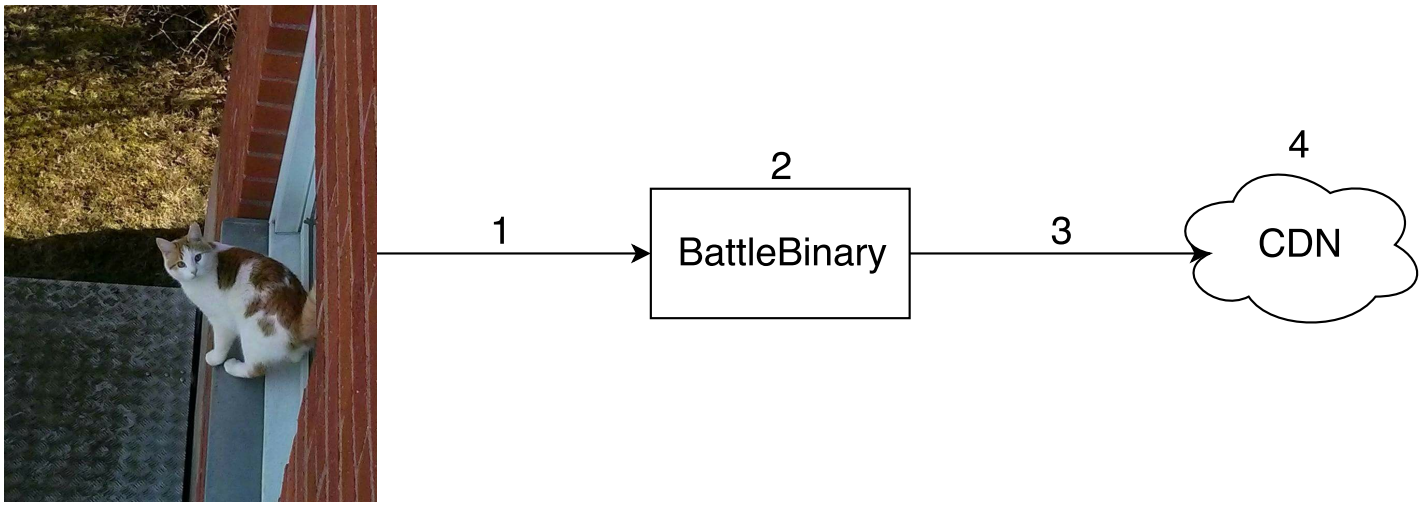
Virtual file system to organise CDN assets



1. Image is uploaded to BattleBinary
2. Filename + Part of file's hash = Asset Identifier
3. Upload image to CDN
4. Image is available to everybody with link `http://ea.akamaihd.net/cat-f1ee0283b6accd6.jpg`

BattleBinary

Virtual file system to organise CDN assets



- 1. Image is uploaded to BattleBinary
- 2. Filename + Part of file's hash = Asset Identifier
- 3. Upload image to CDN
- 4. Image is available to everybody with link `http://ea.akamaihd.net/cat-f1ee0283b6accd6.jpg`

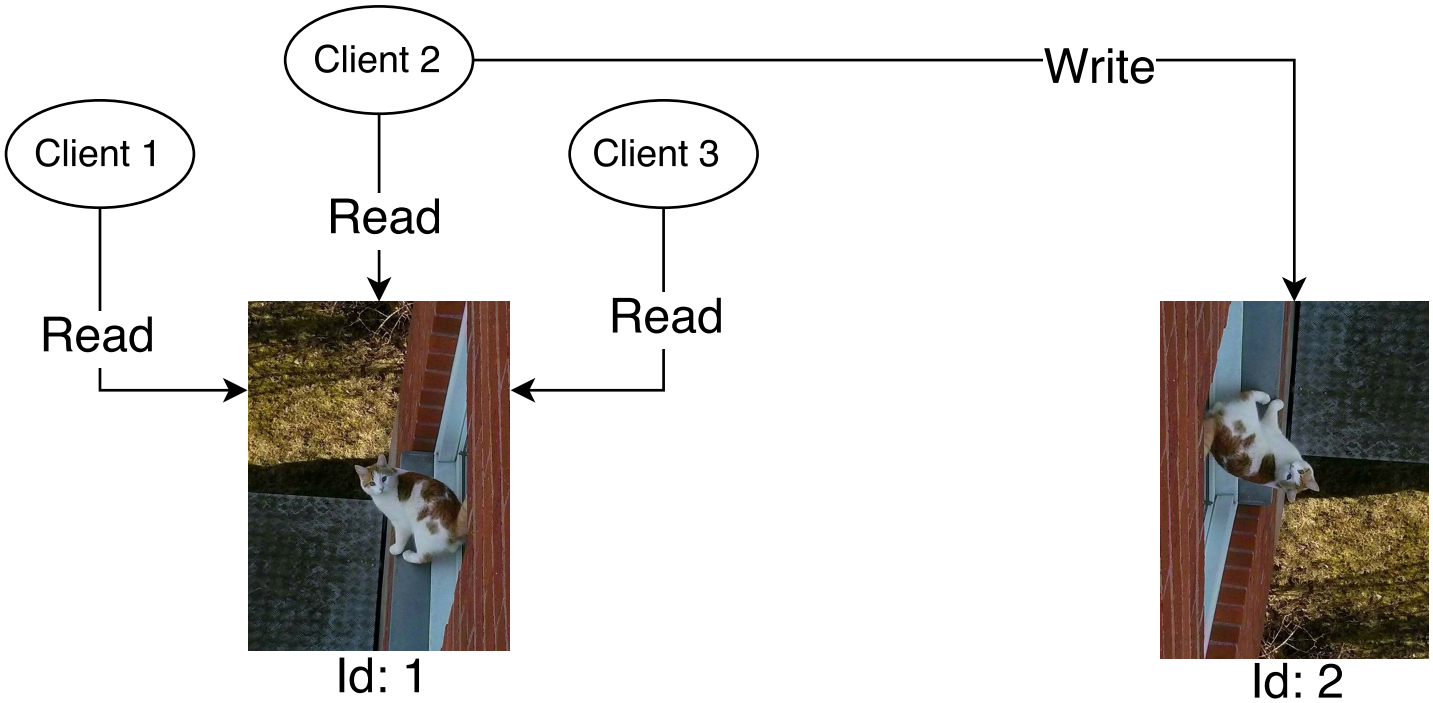
BattleBinary

- A management system for CDN assets

What Uprise wanted:

- Possibility to handle private assets
- Migration to their current technology stack
- Features like snapshots and branching

Copy-on-Write



*IDs are GUIDs in real implementations

Effects of COW

- No accidental incremental changes or race conditions
- No locks needed → Scalability
- Take snapshots of system in constant time
- Needs some form of garbage collection/awareness

Effects of COW

- No accidental incremental changes or race conditions
- No locks needed $\rightarrow$ Scalability
- Take snapshots of system in constant time
- Needs some form of garbage collection/awareness

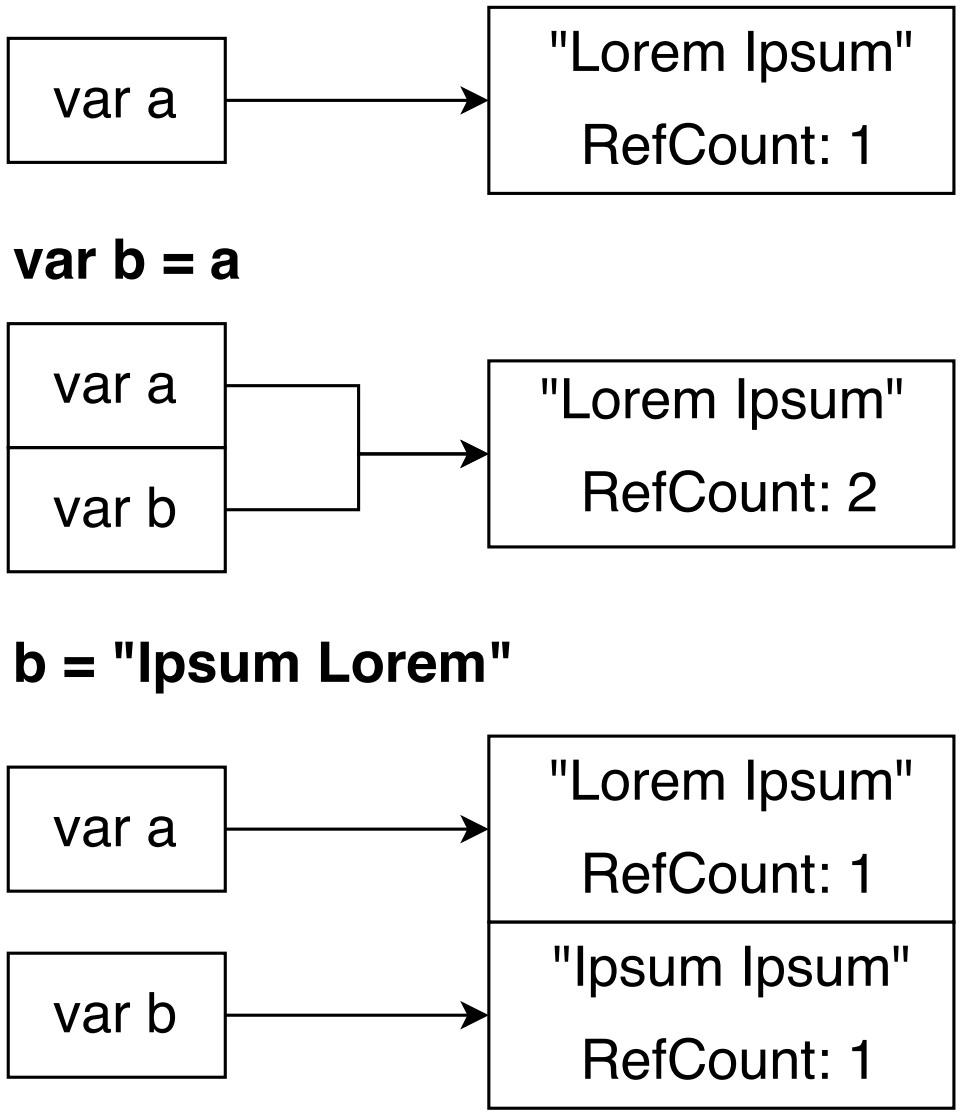
Effects of COW

- No accidental incremental changes or race conditions
- No locks needed $\rightarrow$ Scalability
- Take snapshots of system in constant time
- Needs some form of garbage collection/awareness

Effects of COW

- No accidental incremental changes or race conditions
- No locks needed $\rightarrow$ Scalability
- Take snapshots of system in constant time
- Needs some form of garbage collection/awareness

Garbage


var a
"Lorem Ipsum"
RefCount: 1
var b = a
var a
var b
"Lorem Ipsum"
RefCount: 2
b = "Ipsum Lorem"
var a
var b
"Lorem Ipsum"
RefCount: 1
"Ipsum Ipsum"
RefCount: 1

UPPSALA
UNIVERSITET

Related Work

- Mach kernel
- Btrfs
- Programming Languages

UPPSALA
UNIVERSITET

Related Work

- Mach kernel
- Btrfs
- Programming Languages

UPPSALA
UNIVERSITET

Related Work

- Mach kernel
- Btrfs
- Programming Languages

Problem

- Insecure
- Impractical
- Wont scale
- Lacks necessary features like access control and snapshots

Problem

- Insecure
- Impractical
- Wont scale
- Lacks necessary features like access control and snapshots

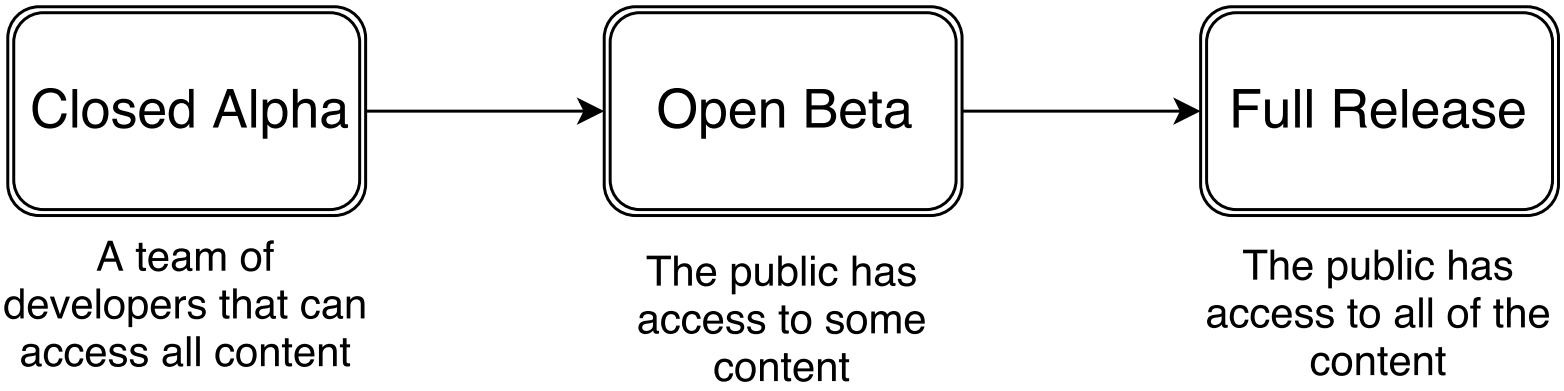
Problem

- Insecure
- Impractical
- Wont scale
- Lacks necessary features like access control and snapshots

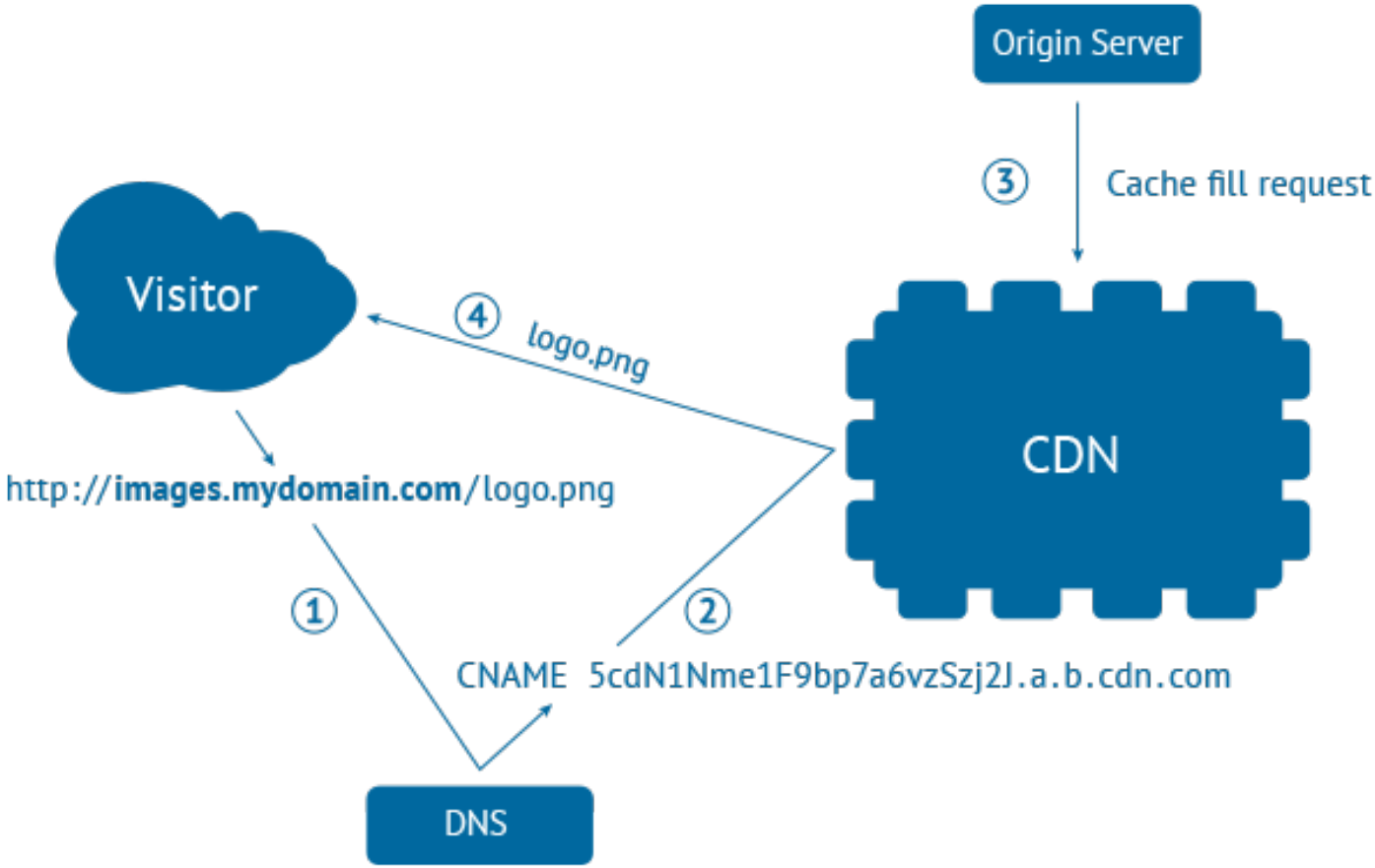
Problem

- Insecure
- Impractical
- Wont scale
- Lacks necessary features like access control and snapshots

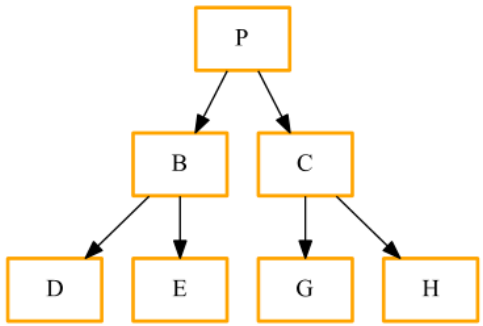
Private Content



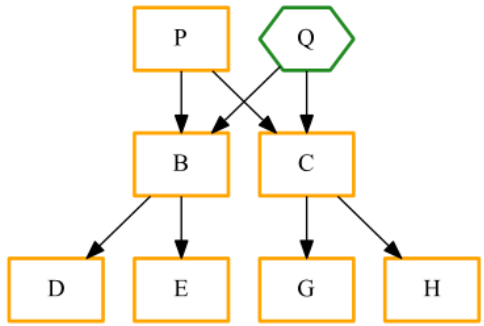
Solutions Idea



Example Operation



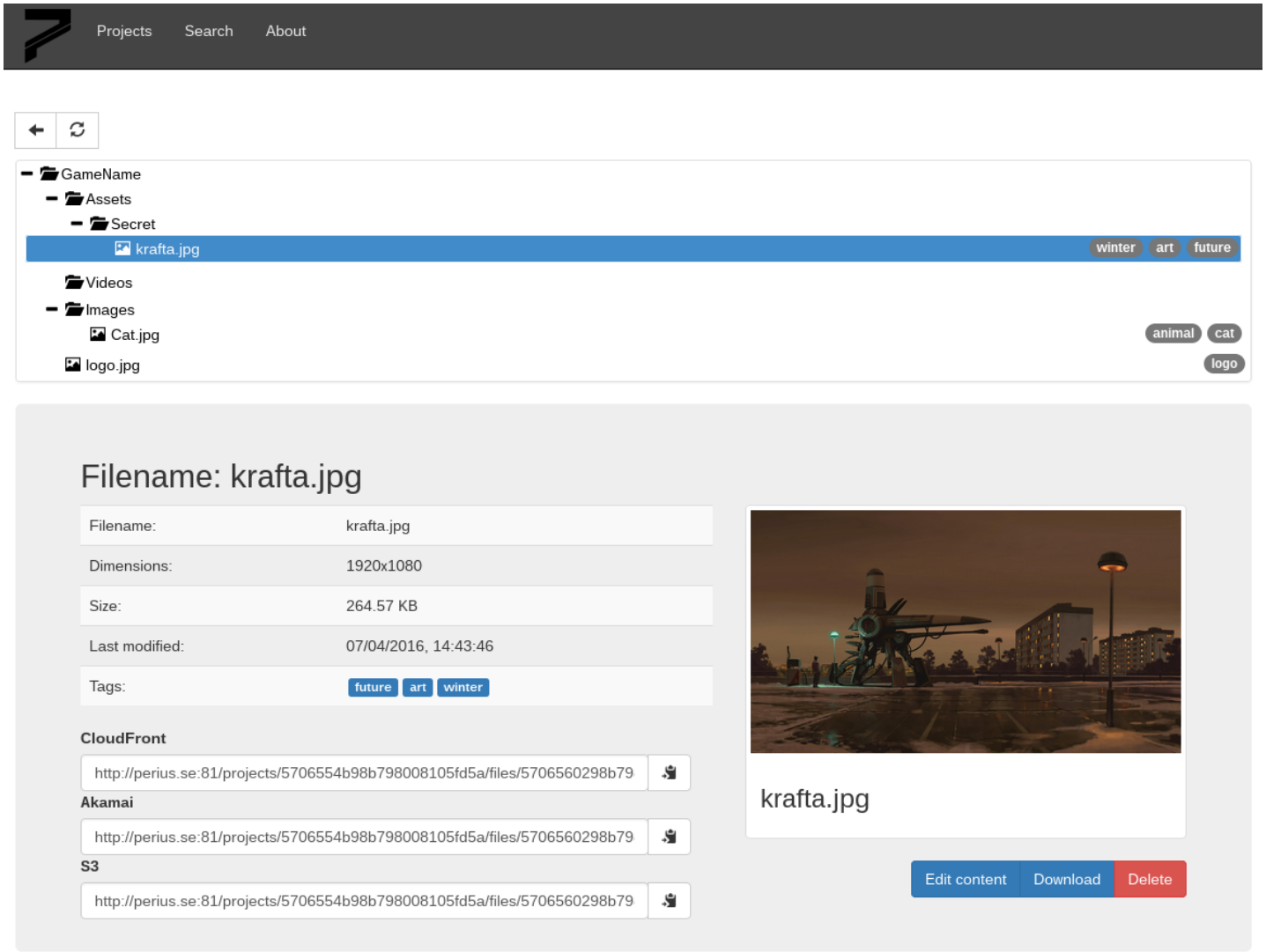
(a) Tree T_p



(b) T_p cloned to T_q

BTRFS: The Linux B-Tree Filesystem, O. Rodeh et al

Virtual Filesystem



Model

- Describes the system that is to be implemented
- Defines the core operations
- Model informally checked through JPF
- Inspired by Biba. et al

Model

- Describes the system that is to be implemented
- Defines the core operations
- Model informally checked through JPF
- Inspired by Biba. et al

Model

- Describes the system that is to be implemented
- Defines the core operations
- Model informally checked through JPF
- Inspired by Biba. et al

Model

- Describes the system that is to be implemented
- Defines the core operations
- Model informally checked through JPF
- Inspired by Biba. et al

Model - Sets

Set	Name	Relation	Type
C	Containers	$\mathbf{C} \supseteq C \ni c$	$\mathbf{C} = P_{fin}(P_{fin}(C \cup M))$
M	Content	$\mathbf{M} \supseteq M \ni m$	$\mathbf{M} = P_{fin}(M)$
F	Files	$\mathbf{F} \supseteq F \ni f$	$\mathbf{F} = P_{fin}(F)$

Model - File Creation

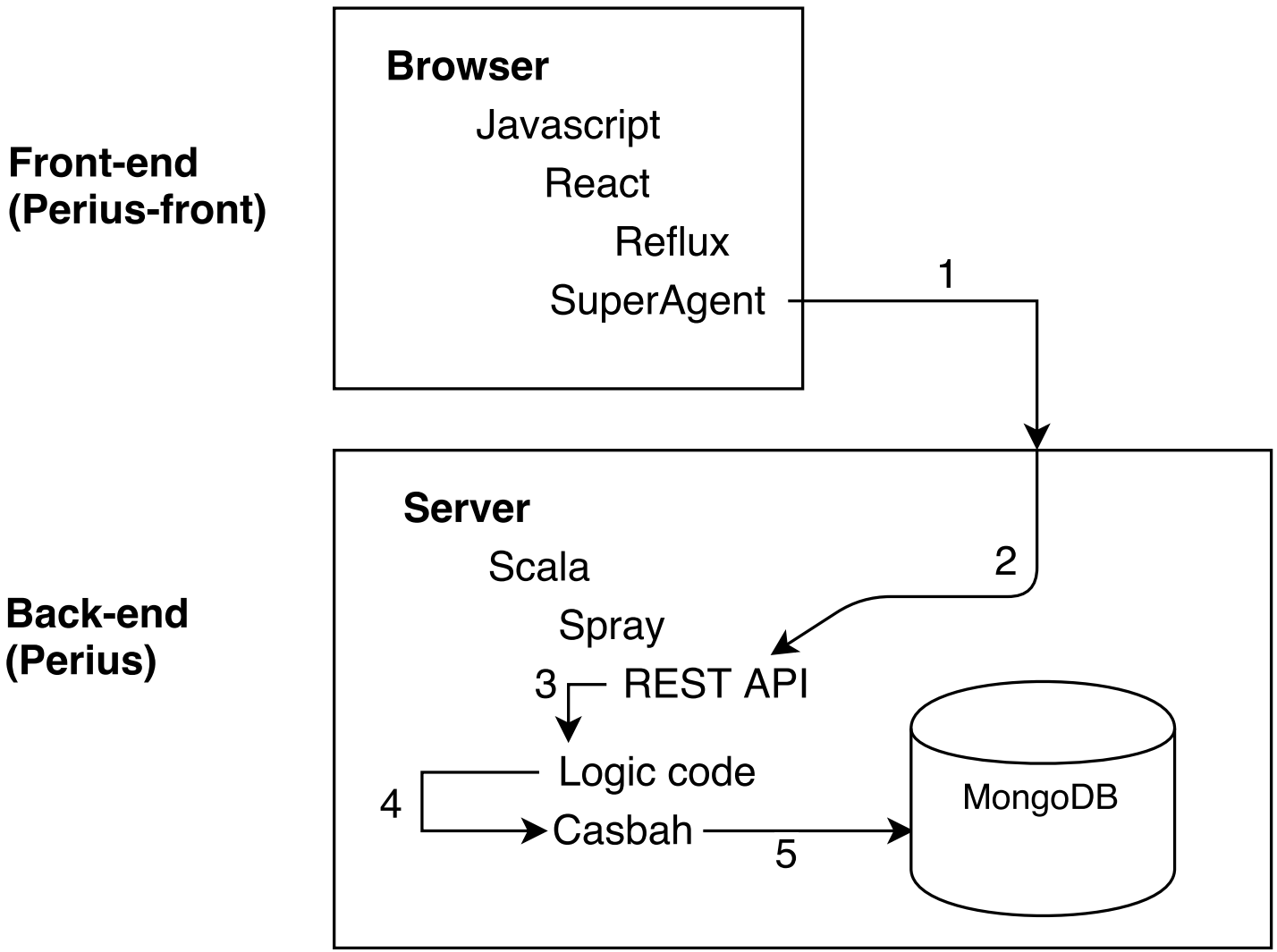
Rule of Inference

$$\frac{C \ni c \qquad A[u,c] \qquad m \notin c \qquad \neg readOnly(c)}{(C \cup \{c\}, F) \xrightarrow{u,create(m,c)} (C \cup \{c \cup \{m\}\}, F \cup \{m.file\})} \quad (1)$$

Persistent Storage

- Unaware of Copy-on-Write
- JSON (sort of) documents stored instead of tables

Stack


Front-end
(Perius-front)
Browser
Javascript
React
Reflux
SuperAgent
1
Back-end
(Perius)
Server
Scala
Spray
2
3
REST API
Logic code
4
Casbah
5
MongoDB

Security Settings

- **Public** - The content can be reached by anybody
- **Protected** - The content can only be reached by a range if IP addresses
- **Private** - The content can only be reached by users with a signed cookie

Security Settings

- **Public** - The content can be reached by anybody
- **Protected** - The content can only be reached by a range if IP addresses
- **Private** - The content can only be reached by users with a signed cookie

Security Settings

- **Public** - The content can be reached by anybody
- **Protected** - The content can only be reached by a range if IP addresses
- **Private** - The content can only be reached by users with a signed cookie

Snapshots (and branching)

(a) Tree T_p

(b) T_p cloned to T_q

Load Testing

- Rigorous Load Testing was done on the back-end
- Wrk and Apache Bench was used
- Result: About 60000 clients/back-end node before congestion

Load Testing

- Rigorous Load Testing was done on the back-end
- Wrk and Apache Bench was used
- Result: About 60000 clients/back-end node before congestion

Load Testing

- Rigorous Load Testing was done on the back-end
- Wrk and Apache Bench was used
- Result: About 60000 clients/back-end node before congestion

Scalability

- Unlimited scaling on width
- Optional Non-Blocking I/O with Reactive Mongo
- Some heavy operations can be solved with caching

Scalability

- Unlimited scaling on width
- Optional Non-Blocking I/O with Reactive Mongo
- Some heavy operations can be solved with caching

Scalability

- Unlimited scaling on width
- Optional Non-Blocking I/O with Reactive Mongo
- Some heavy operations can be solved with caching

Open Source

- https://github.com/spydon/perius
- https://github.com/spydon/perius-front

UPPSALA
UNIVERSITET

Questions?

UPPSALA
UNIVERSITET

Thank you for listening