Privacy Enforcement for Untrusted Cloud Applications

Emil Stefanov (UC Berkeley)

http://www.emilstefanov.net/

with

Krste Asanovic
UC Berkeley

Jin Chen
Fudan University

David Culler
UC Berkeley

Petros Maniatis
Intel

Prashanth Mohan
UC Berkeley

Charalampos Papamanthou
UC Berkeley

Elaine Shi
University of Maryland

Dawn Song
UC Berkeley

Mohit Tiwari
UC Berkeley

Nguyen Tran
Google

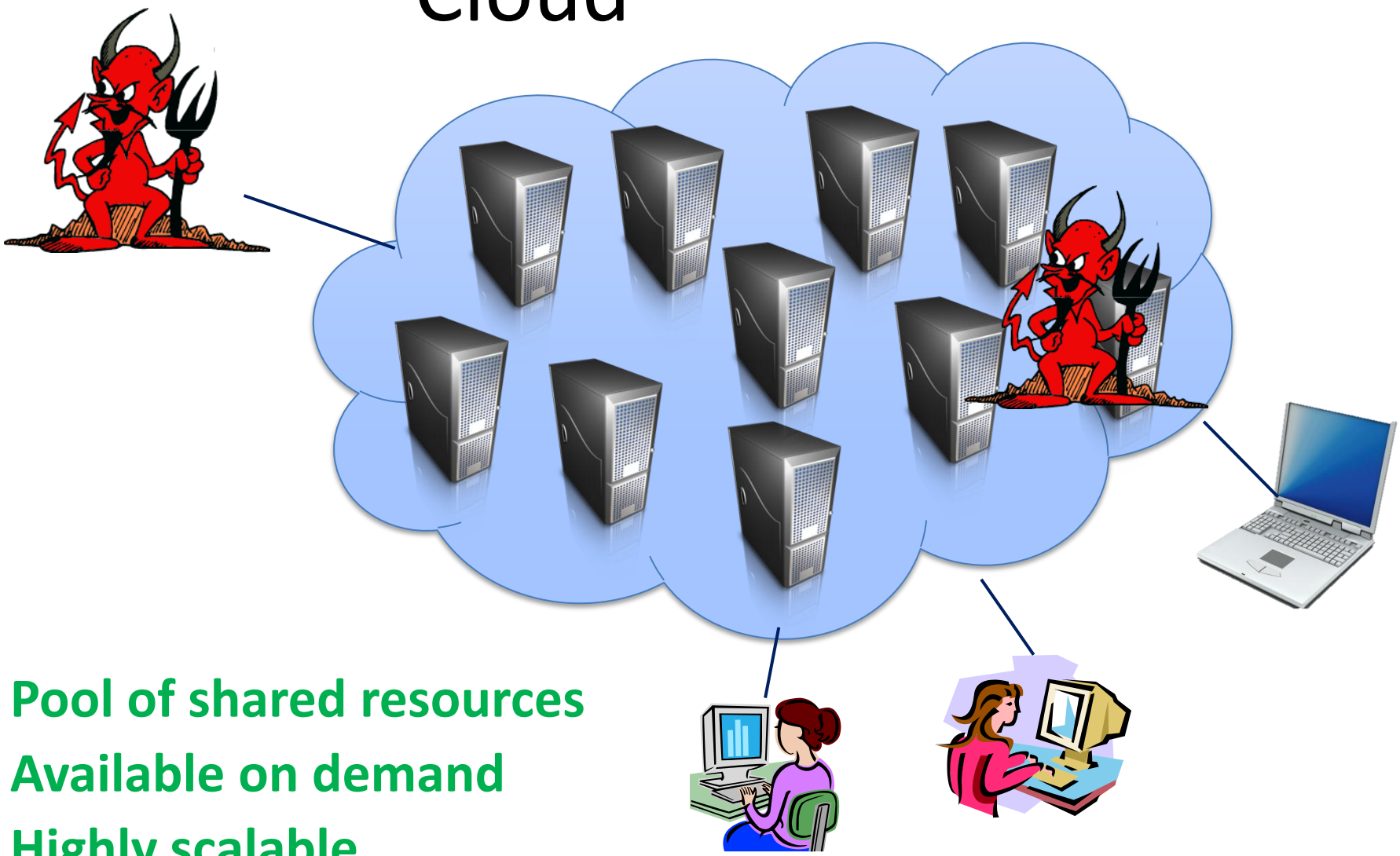
Special thanks to Mohit Tiwari for providing some slides.

The Team

The Age of Big Data

Plentiful, and Private

Cloud



- Pool of shared resources
- Available on demand
- Highly scalable
- Lack of control & visibility

Need Data Protection as a Service

Vulnerable software

TODAY @ PCWORLD

RockYou Sued Over Data Breach

By Brennon Slattery, PCWorld Dec 31, 2009 8:40 AM

(Un) Intentional Misuse

FRIDAY, MARCH 12, 2010

Netflix Prize Update

In light of all this, we have decided to not pursue the Netflix Prize sequel that we announced on August 6, 2009.

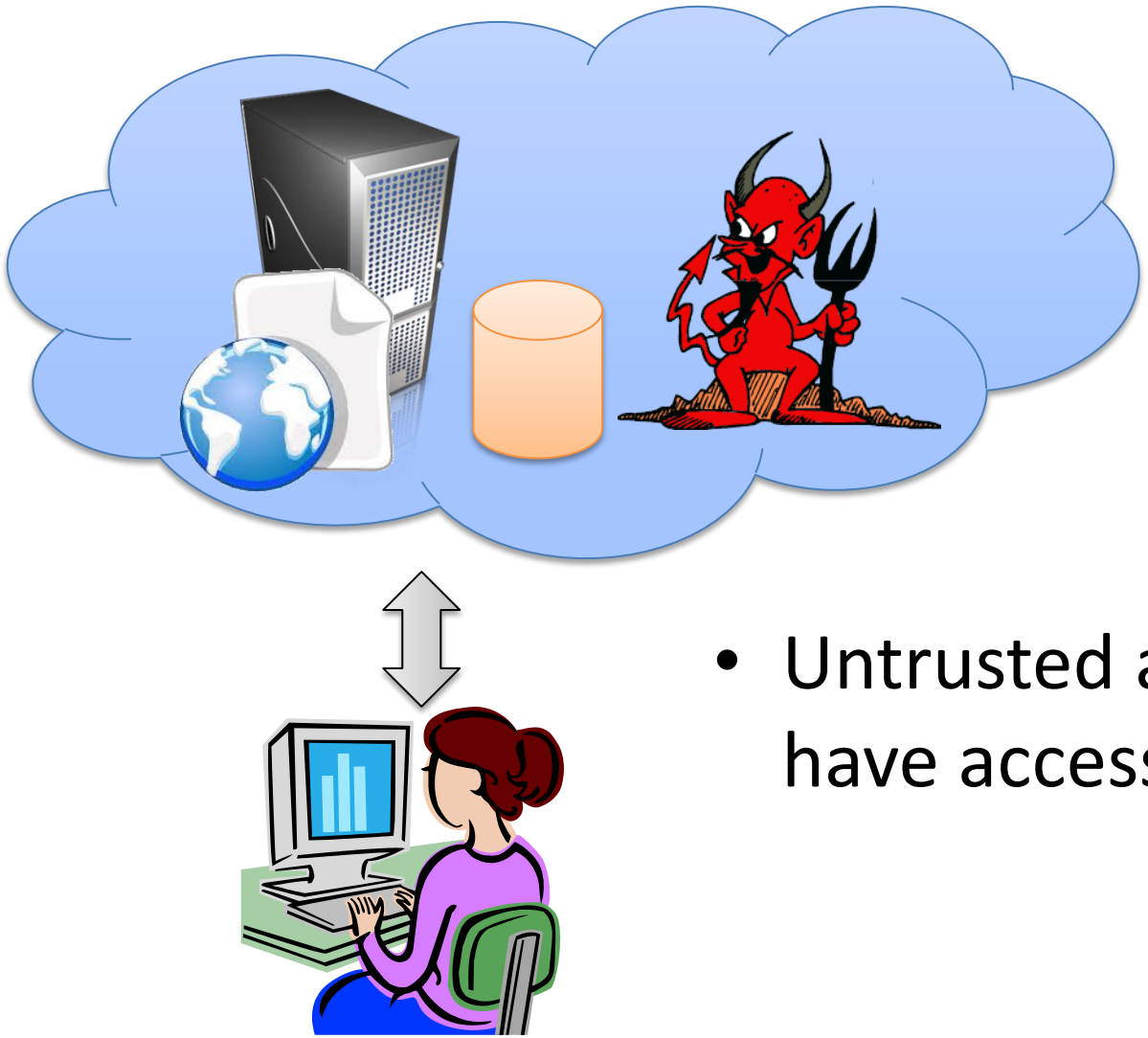
Insider Attacks

Google fires employee for snooping on users

The Internet search giant says the software engineer broke its 'strict internal privacy policies.' He allegedly accessed information about four teenagers.

September 16, 2010 | By Jessica Guynn, Los Angeles Times

Challenge #1



- Untrusted applications have access to user's data.

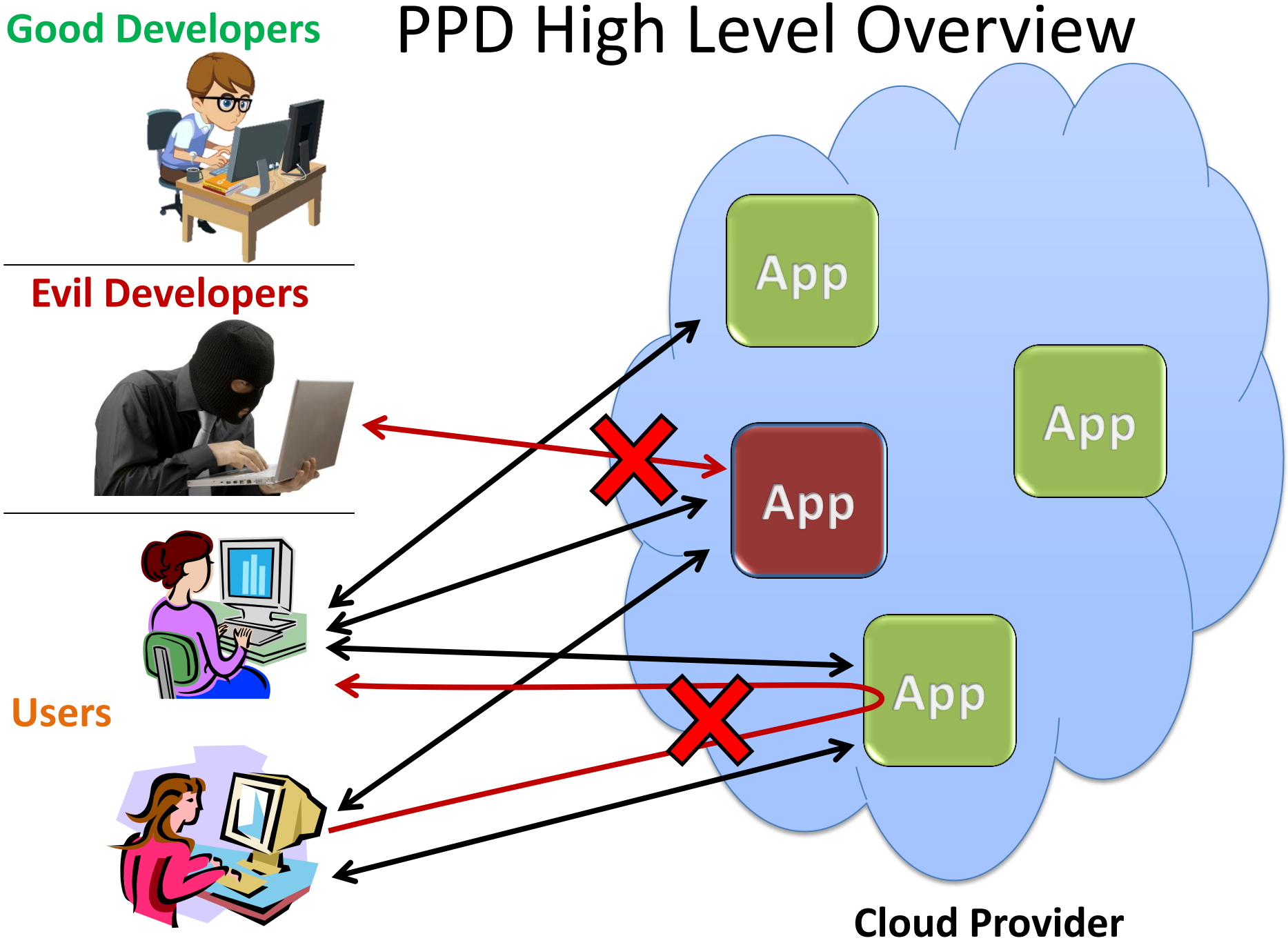
Challenge #2

Novice Users

Challenge #3

- Decentralized:
  - Data
  - Authentication
  - Authorization
  - Identity Management

PPD High Level Overview



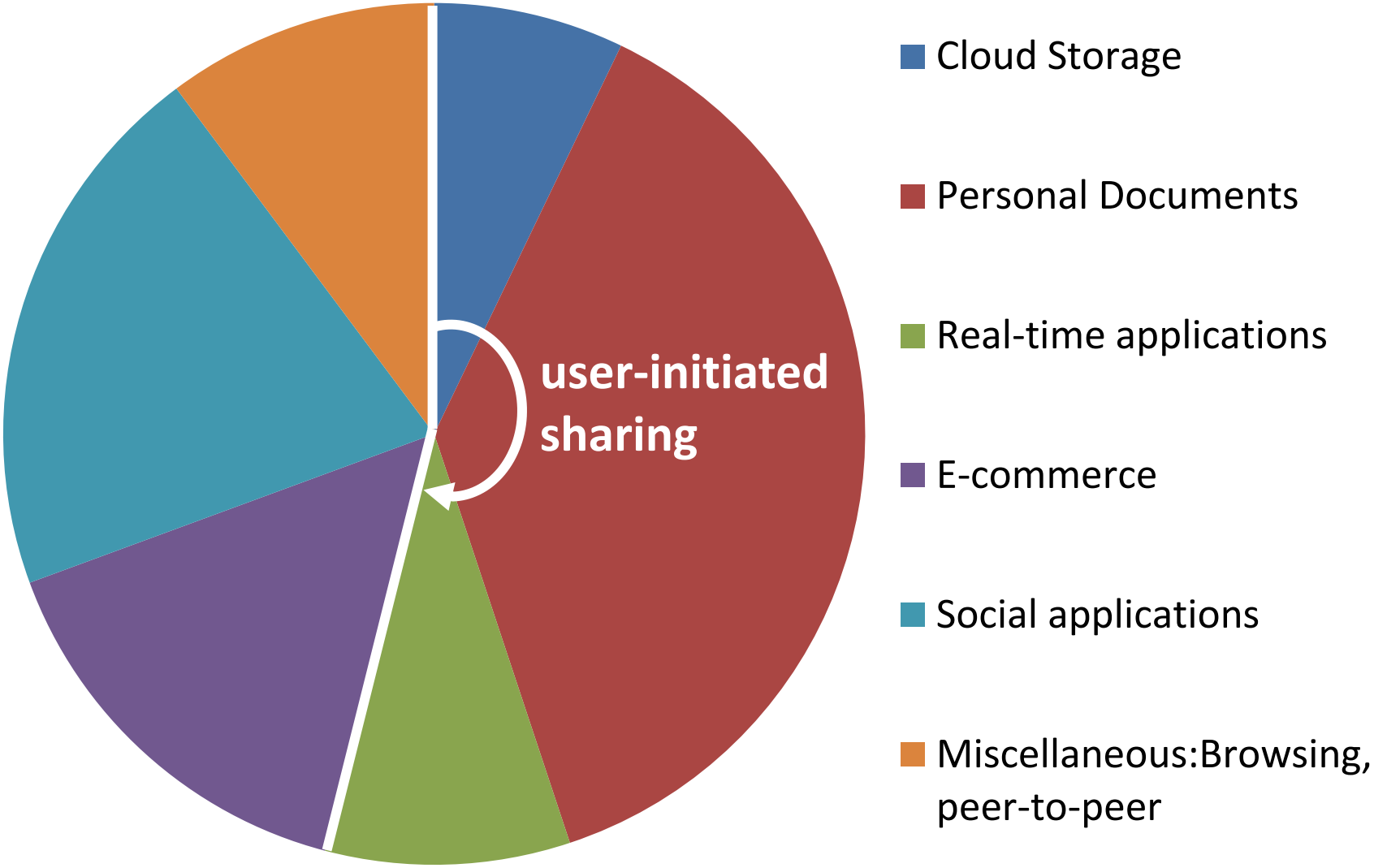
Basic Components

- **Application Instances**
  - Applications are split into instances.
  - Ex: document, photo, chatroom, GIT repository.
- **Data Capsules**
  - One data capsule per application instance.
  - For non-volatile data.
- **Individual Containers**
  - Each instance is isolated: placed inside container.
- **Aggregate Containers**
  - Have read-only access to multiple data capsules.

Access Control

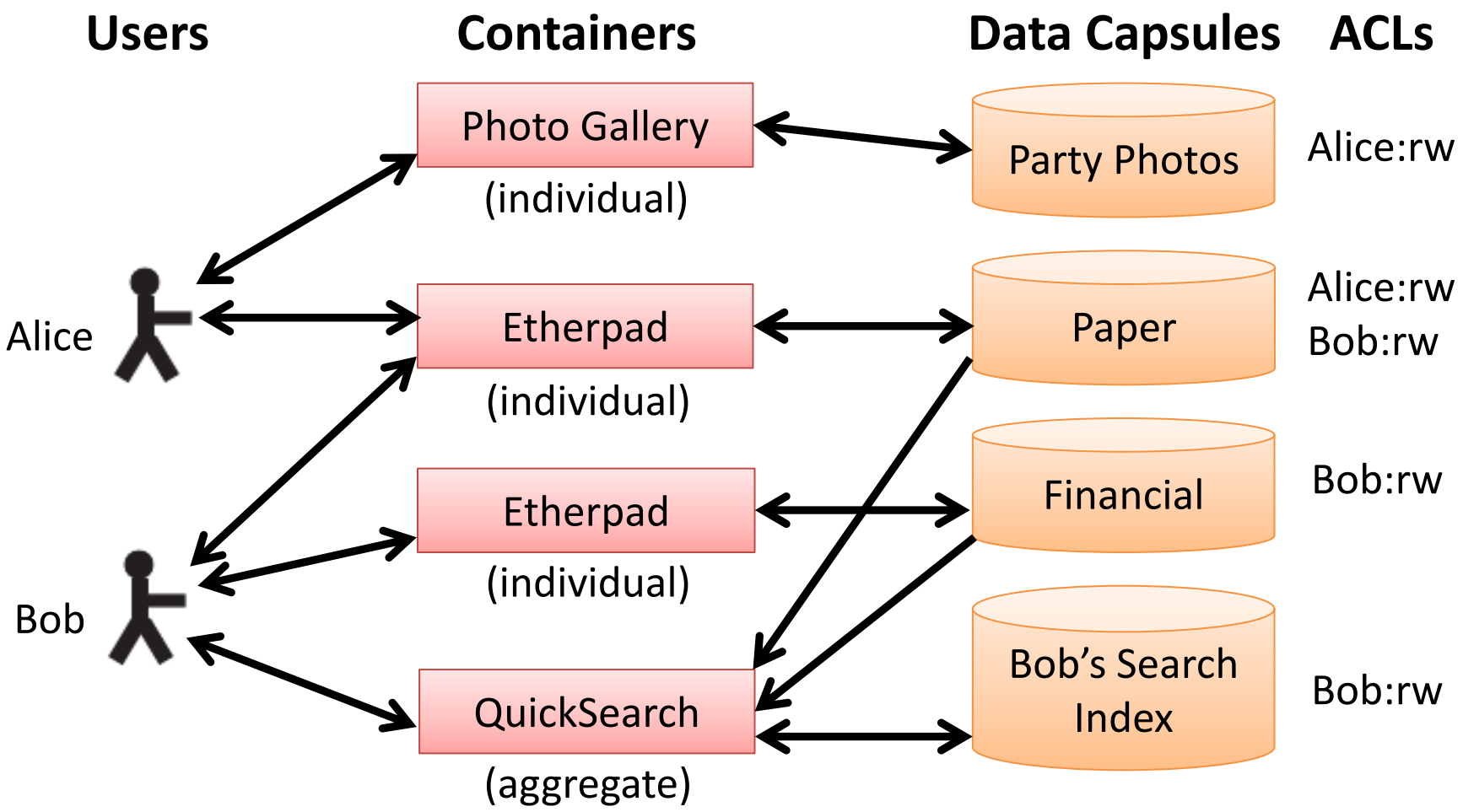
- **Data Capsules have ACLs**
  - { Alice:rw, Bob:rw, Carol:rw, … }
- **Initial ACL**
  - Creator can read/write.
  - If Alice creates a data capsule, the ACL is:
    - { alice:rw }
- **User-initiated sharing**
  - Users can change ACLs
  - Standardized across all apps

100 Top Web Applications



Source: CNET Webware 100 Winners http://www.cnet.com/100/

Example Information Flow



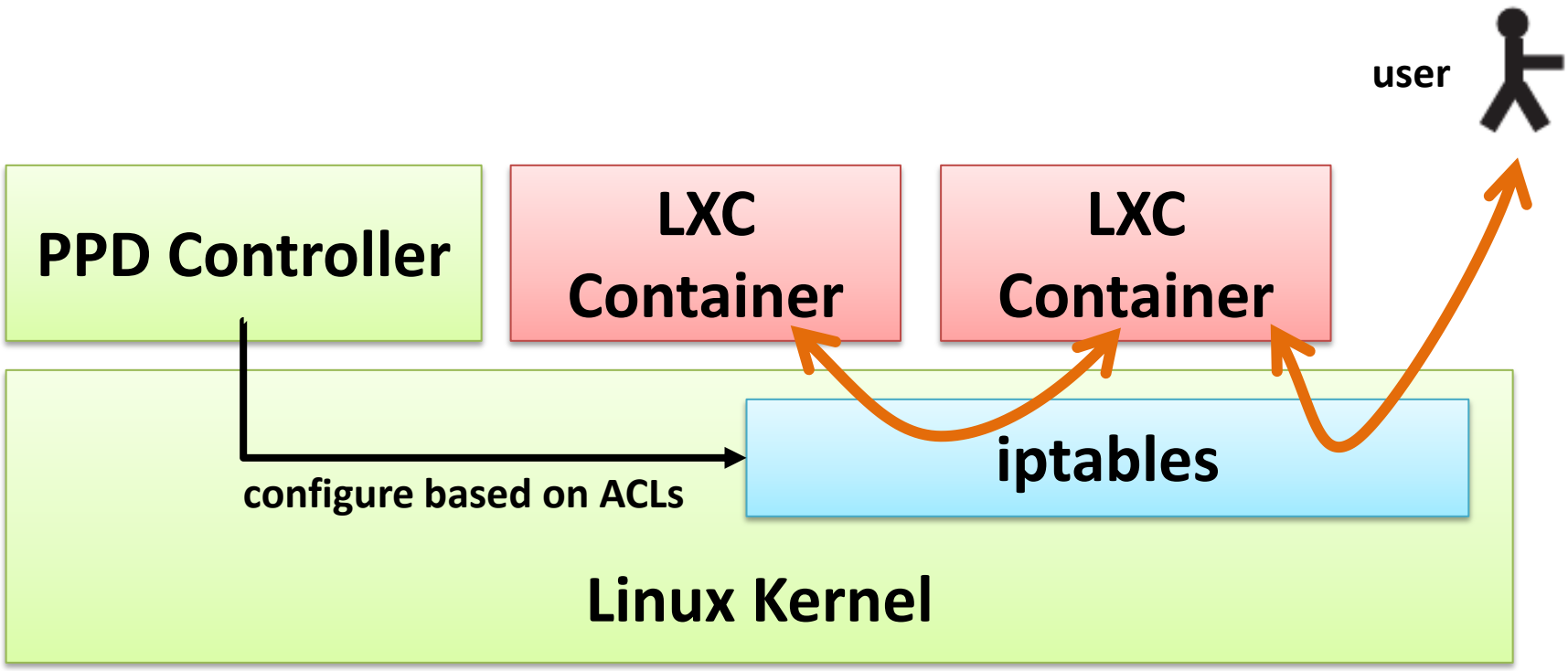
Compatibility

- Want maximum compatibility.
- Legacy applications
  - Very few changes to port.
- Currently supports Linux applications
  - Including web applications running under Apache
- Techniques are generalizable to other operating systems.

LXC Containers

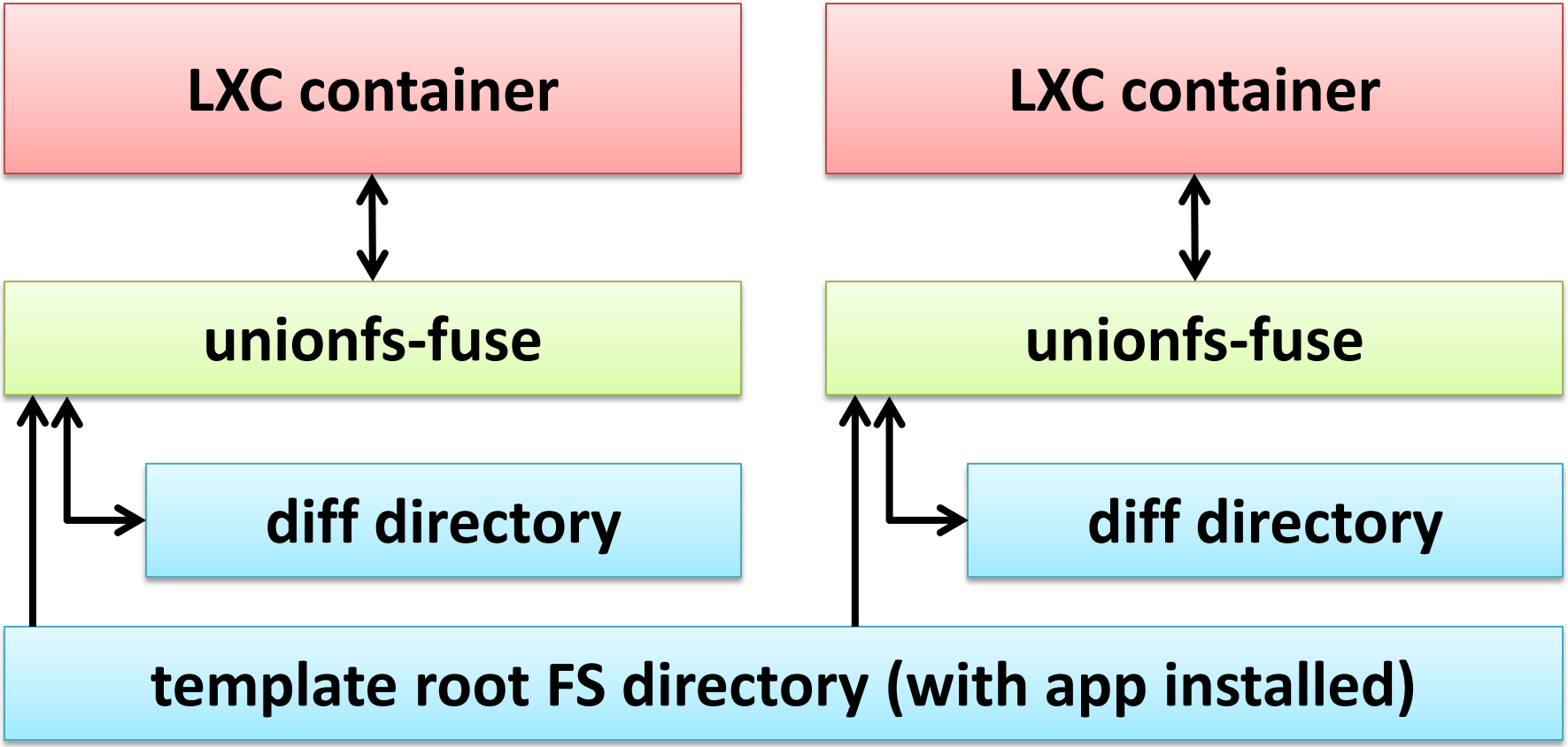
- Like lightweight Linux VM's
- Share common Linux kernel
- Different namespaces
  - E.g., PIDs, network interfaces
  - Further isolation via cgroups
- Boots from subdirectory containing a Linux root directory.
- http://lxc.sourceforge.net/

ACL Based Communication



- ACLs transformed into firewall rules

File System



- Each container has its own CoW root file system.
  - Saves lots of space & improves caching.
  - Container is chroot'ed into its CoW directory.
- Fuse based (more stable than kernel based unionfs)

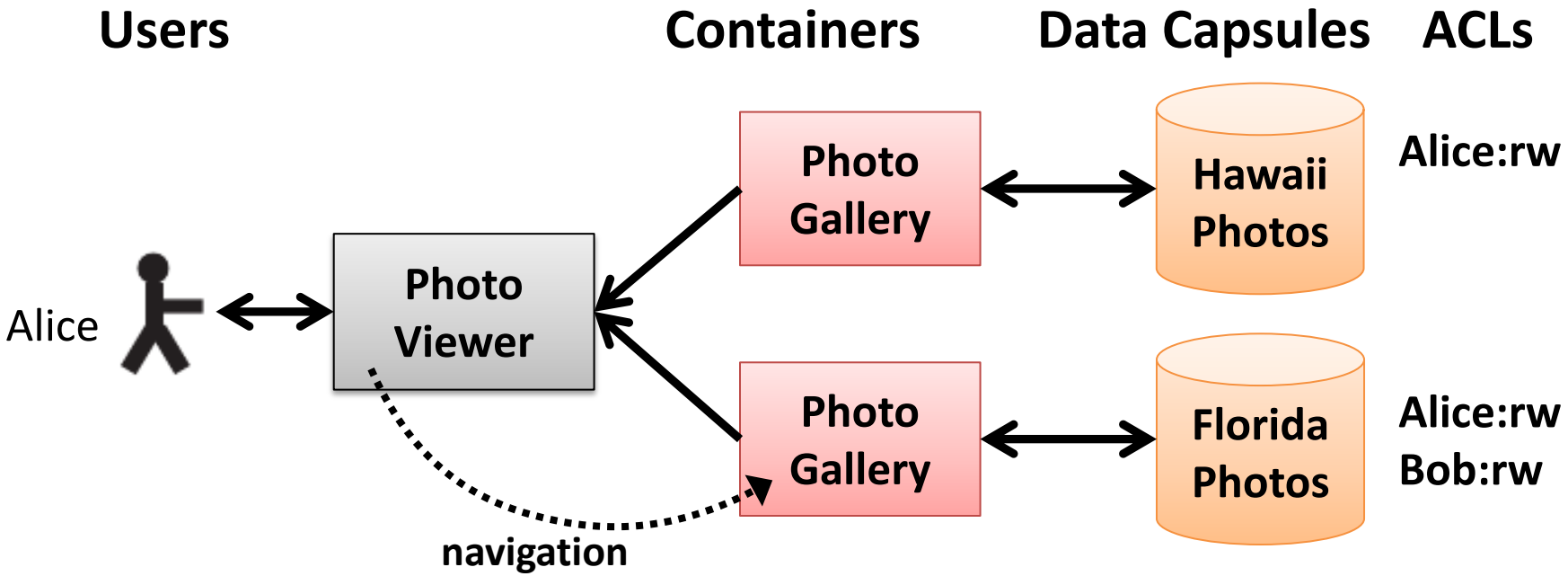
LXC Container Life Cycle

- Too many containers slow down the system (e.g., more than 1,000)
- Need to shut down and recycle containers.
- Need to sync to disk first.
- Applications must specify when they are doing background work.

Viewers

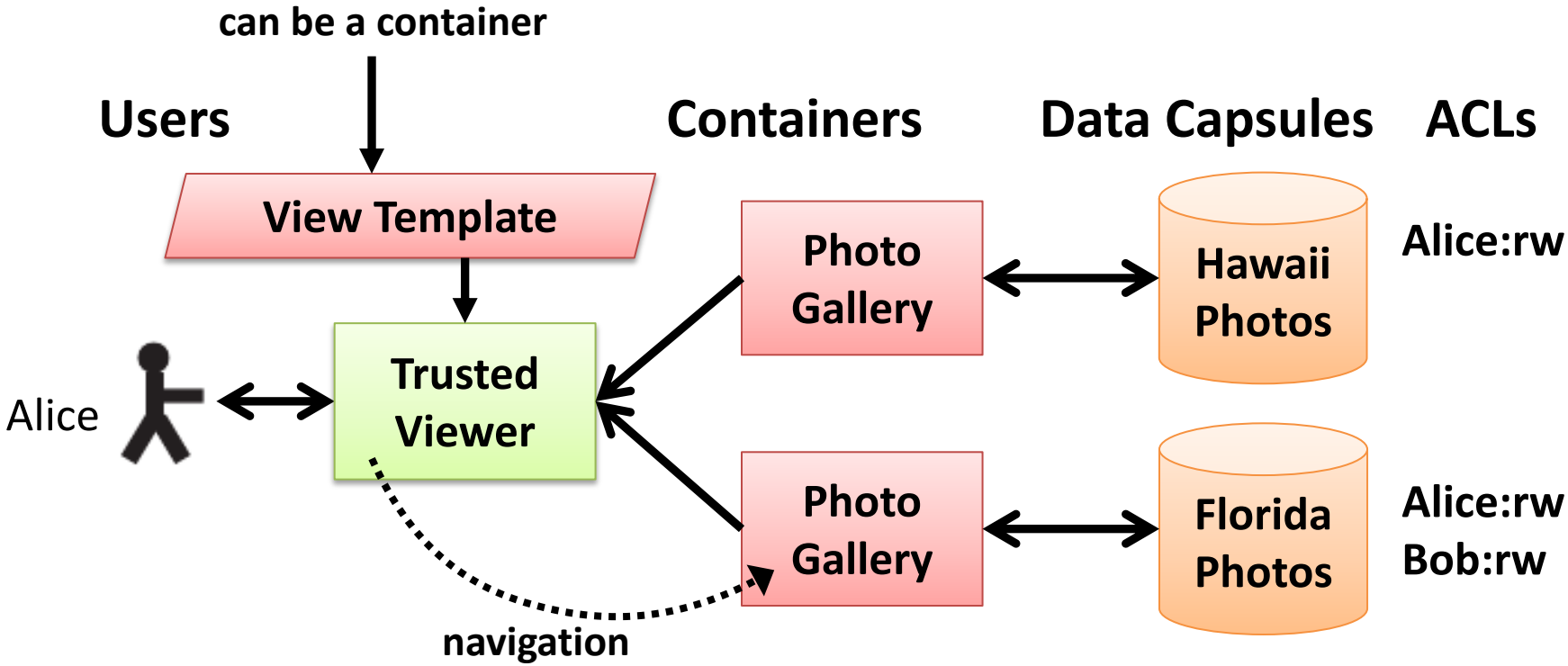
- Problem:
  - Data spread across many containers
- Want to aggregate
  - Search
  - Summary
  - Explore efficiently
- Want to still enforce ACLs
- Our solution:
  - Viewers

Ideal Viewer



- Example: Aggregates search results
- Allows the user to navigate to a search result
- Want:
  - Generic (not application specific)
  - Privacy preserving (ACLs enforced)
  - Query privacy (Alice's query does not leak to Bob) – not covered

PPD Viewer



- Untrusted view template specifies layout
- Trusted viewer with client-side isolation does aggregation
  - Cross-origin (cross-container) isolation in browser
  - Cross-origin navigation prohibited.

PPD Viewer Example

Outer view (static HTML)

Specified in the application manifest

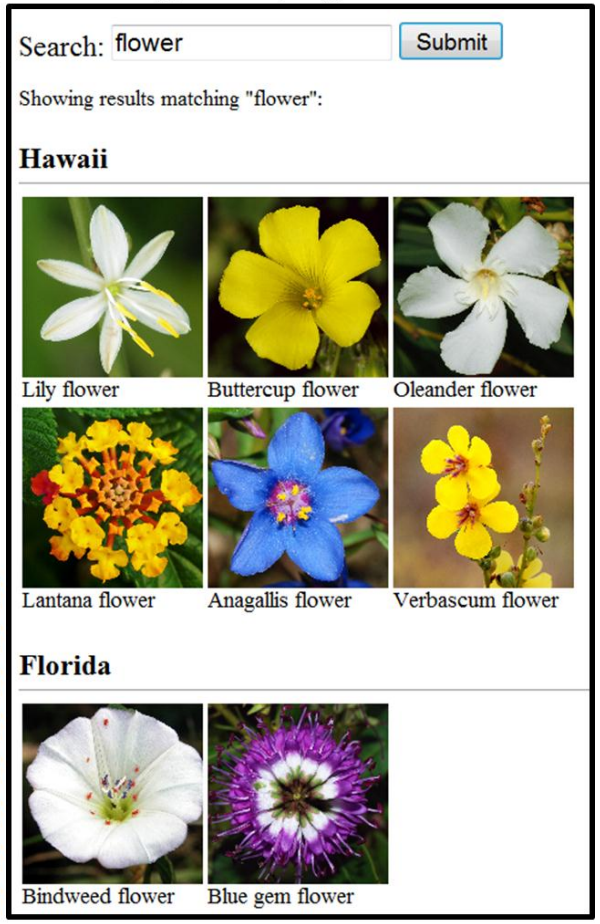
```
<html>
...
Search: <input name="Query“ ... />
<input type="submit“ ... />
...
<p>Showing results matching
"_PPD_Query":</p> ...
<div id="Container1">        </div>
<div id="Container2">        </div>
...
</html>
```

Inner view for container 2 (static HTML)

Generated by container 2

```
<table>
<tr>
<td><a href="#_PPD_Bindweed.jpg">
   <img src="Bindweed.jpg" />
   </a><br/>Bindweed flower</td>
<td><a href="#_PPD_BlueGem.jpg">
   <img src=“BlueGem.jpg“ />
   </a><br/>Blue gem flower</td>
</tr>
</table>
```

Resulting web page



Untrusted Backend Services

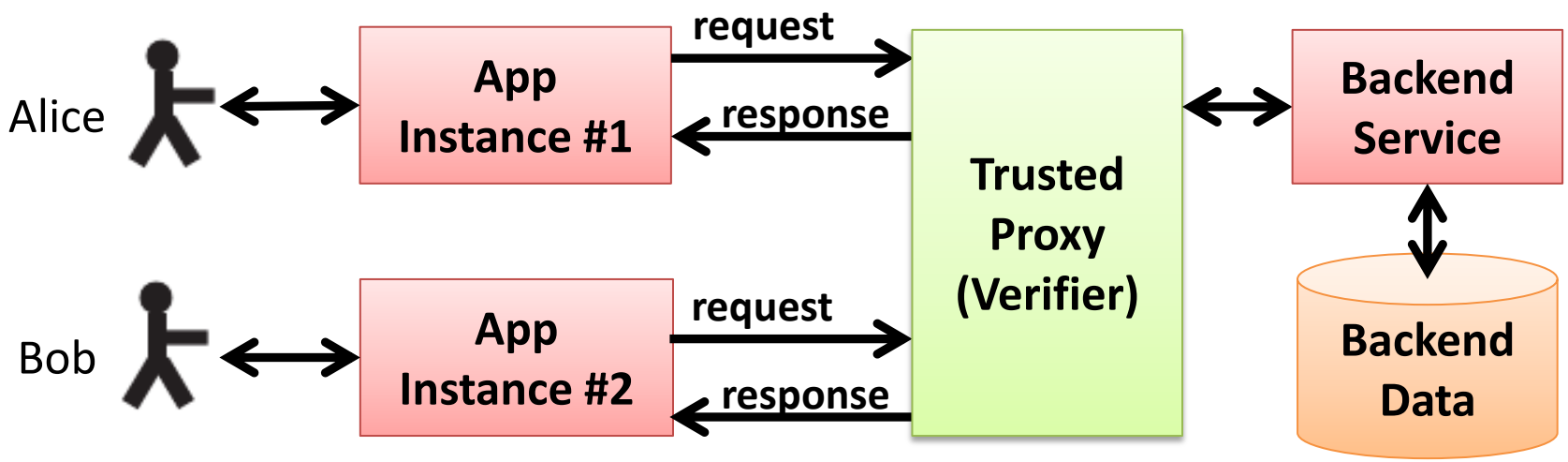
Overview:

- Developers create untrusted backend services.
- Other developers can use them.
- Developers don’t have to trust each other.

Benefit:

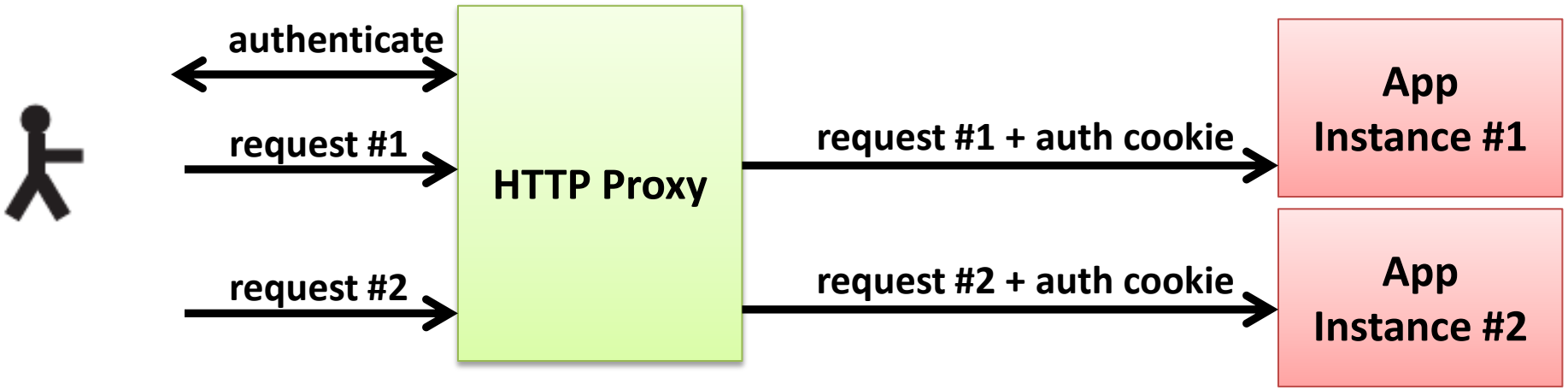
- Decrease overhead while maintaining security.

Verifiable Untrusted Backend Services



- Verifiable services:
  - File system
  - Key-value store
- Uses on-the-fly cryptographic integrity checking.
- Untrusted backend service sees plaintext
  - Deduplication, compression
- Reduced TCB

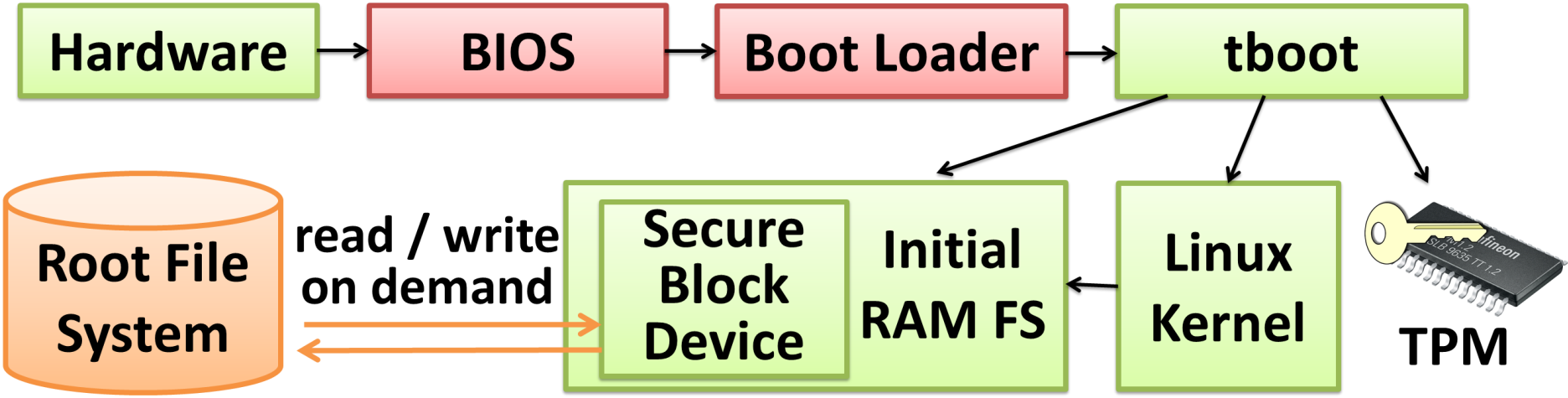
Centralized Authentication



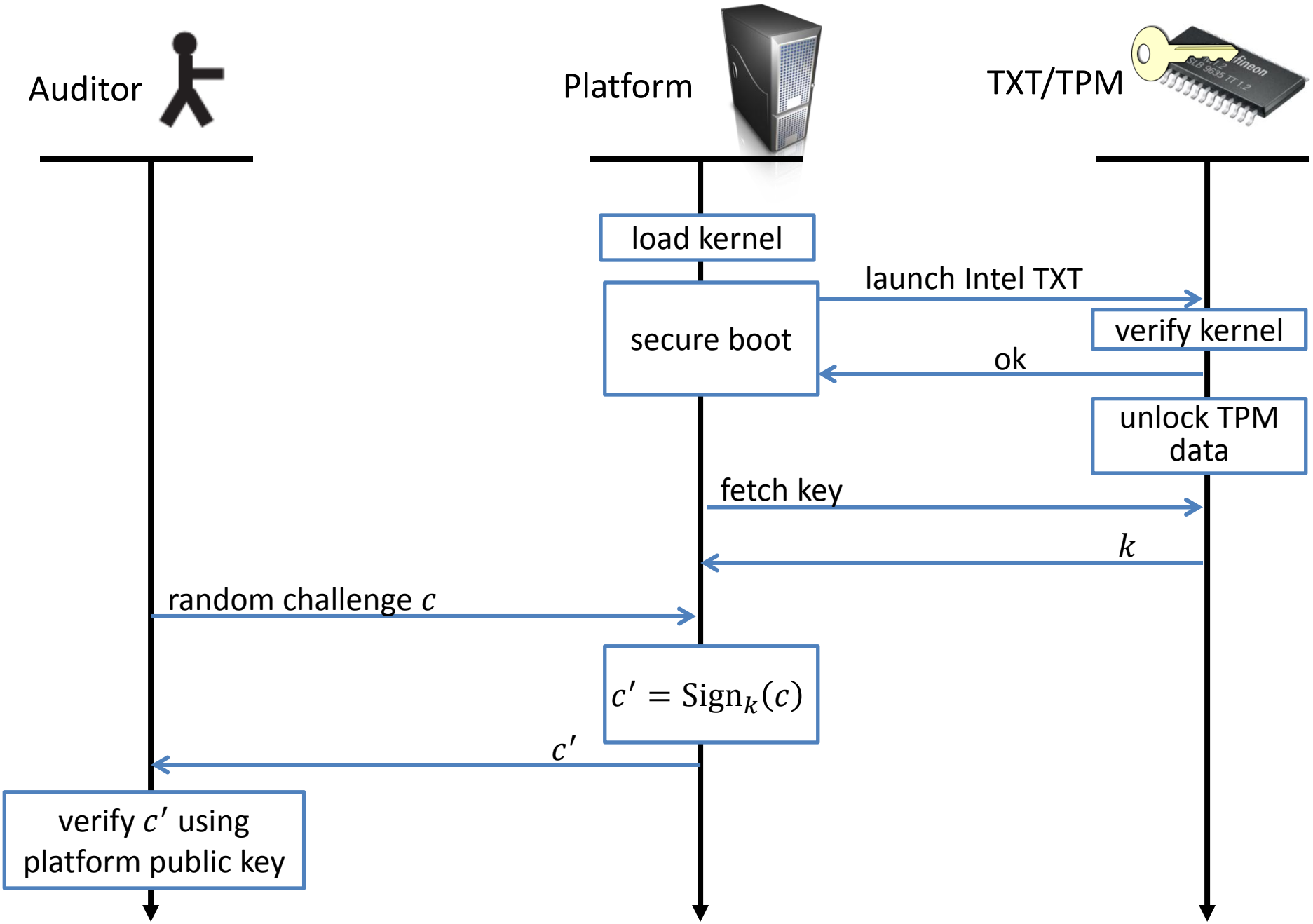
- User authentication is delegated to PPD
- Application just receives auth cookie
  - i.e., authenticated username

Platform Integrity Verification

- Maliciously modified platform will not boot.
- User data can only be decrypted by the verified platform (uses Intel TXT & TPM).



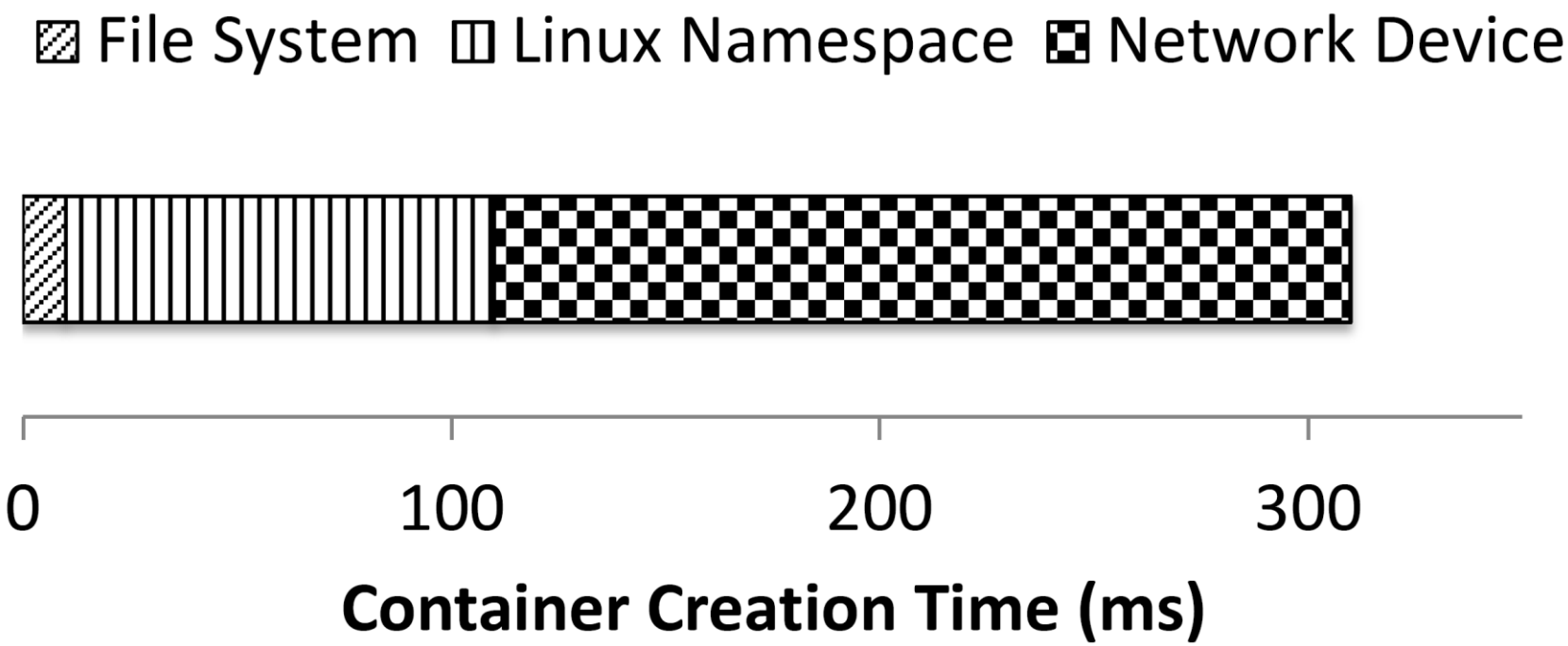
Remote Attestation



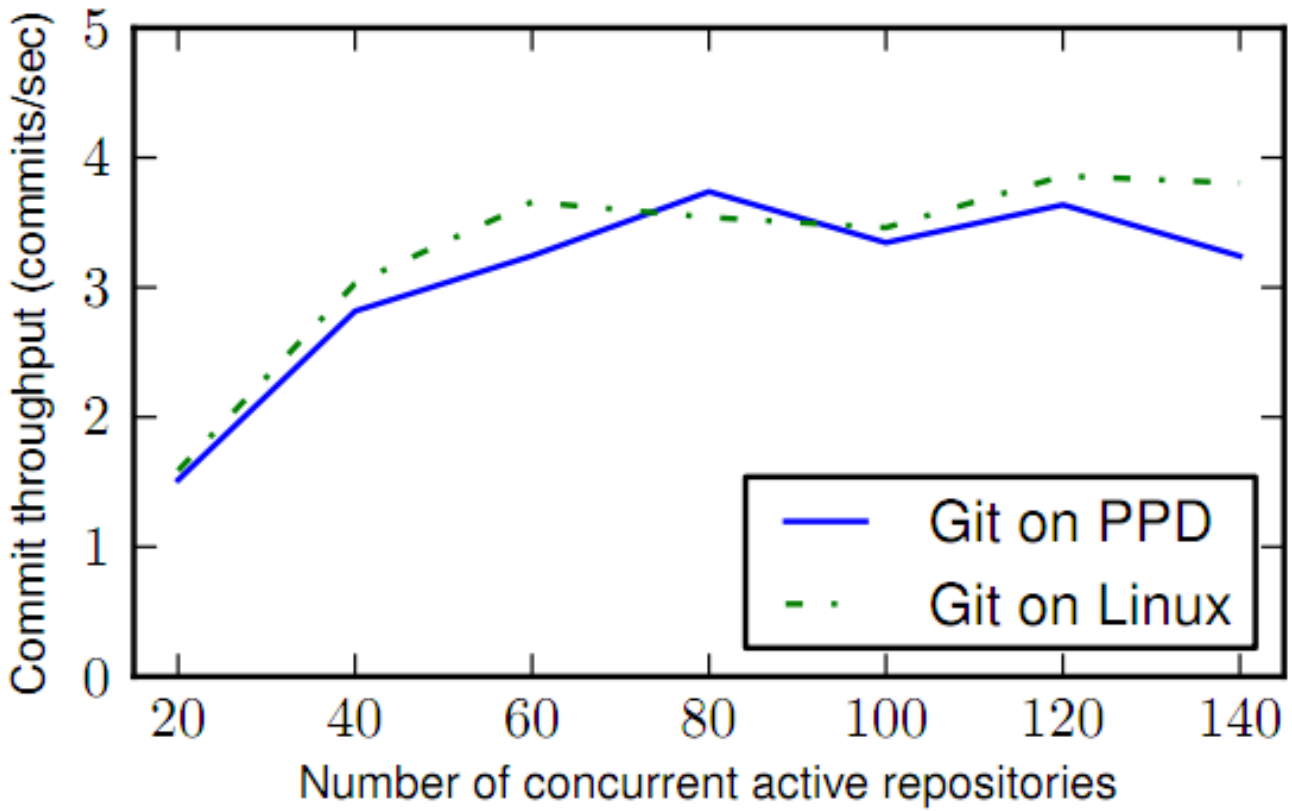
Evaluation

- Micro benchmarks
  - LXC Startup
- Applications
  - GIT
  - Etherpad
  - Friendshare (custom written)

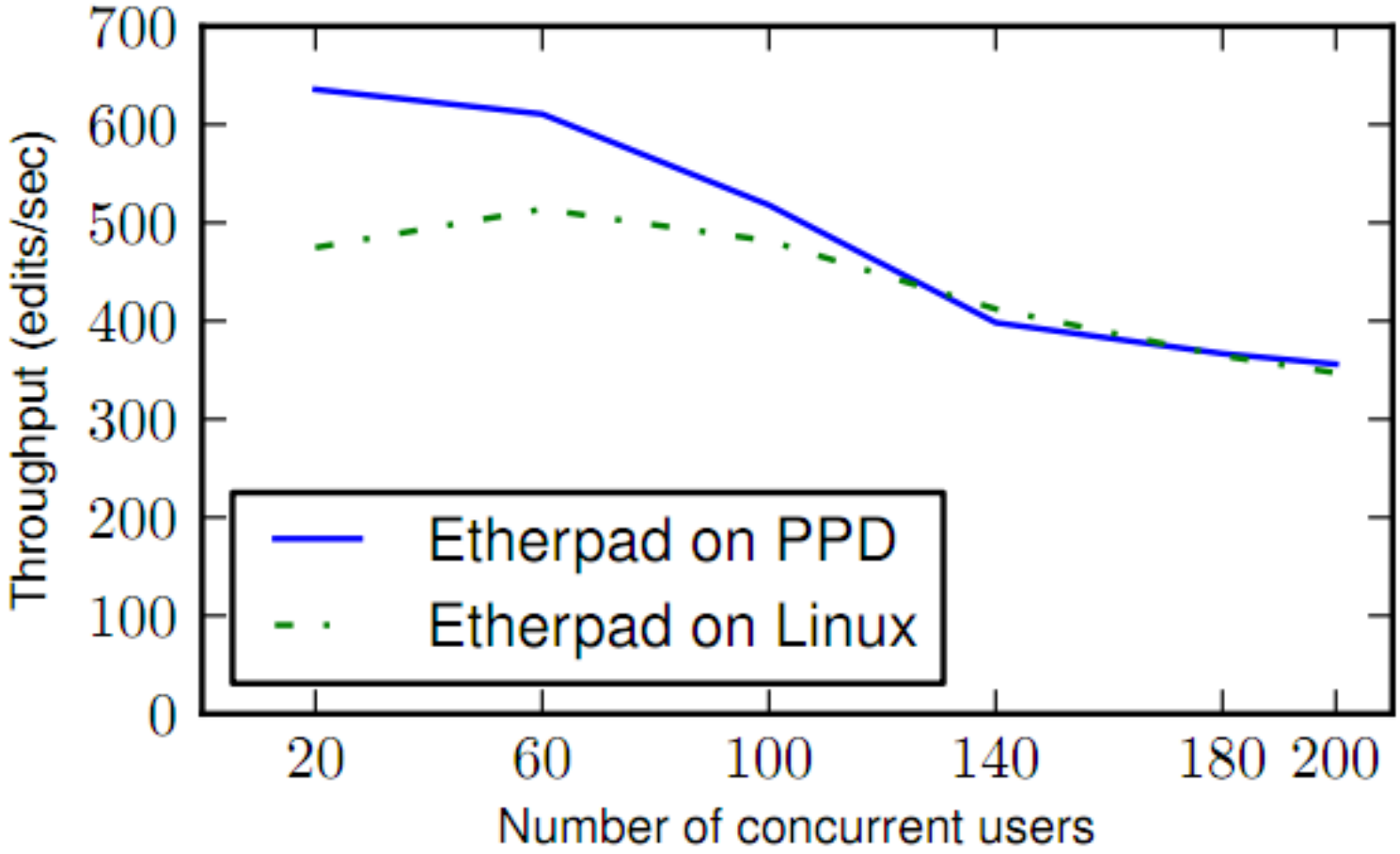
LXC Startup Performance



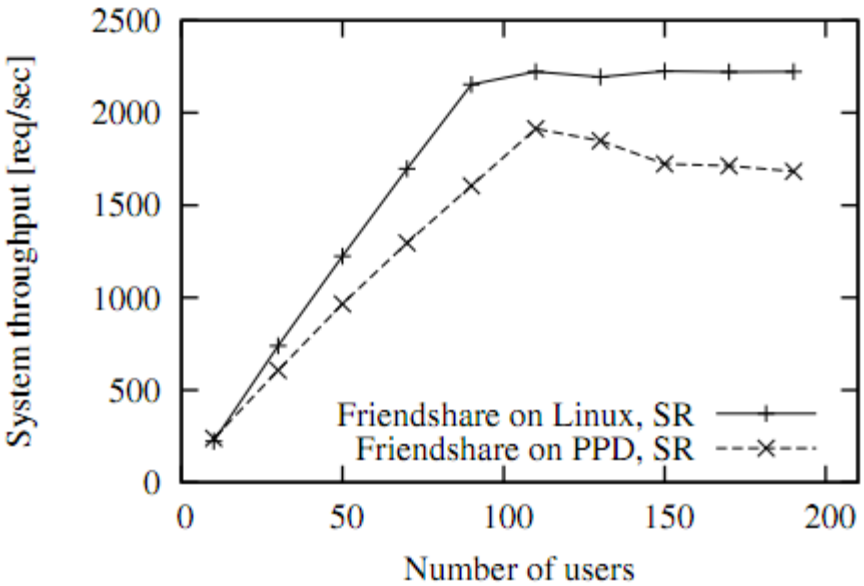
GIT


Commit throughput (commits/sec)
0
1
2
3
4
5
20
40
60
80
100
120
140
Number of concurrent active repositories
Git on PPD
Git on Linux

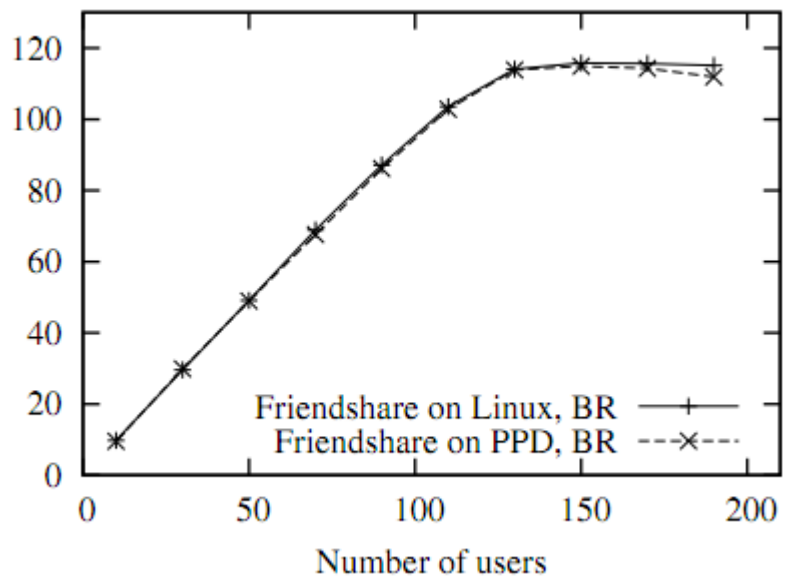
Etherpad


700
600
500
400
300
200
100
0
Throughput (edits/sec)
20
60
100
140
180
200
Number of concurrent users
Etherpad on PPD
Etherpad on Linux

Friendshare Throughput



Small Requests: 10 filenames



Big Requests: 10KB images

Friendshare Latency