Bremen

Scalable Concurrency Control for Massively Collaborative Virtual Environments

Patrick Lange, Rene Weller, Gabriel Zachmann

University of Bremen, Germany

cgvr.cs.uni-bremen.de

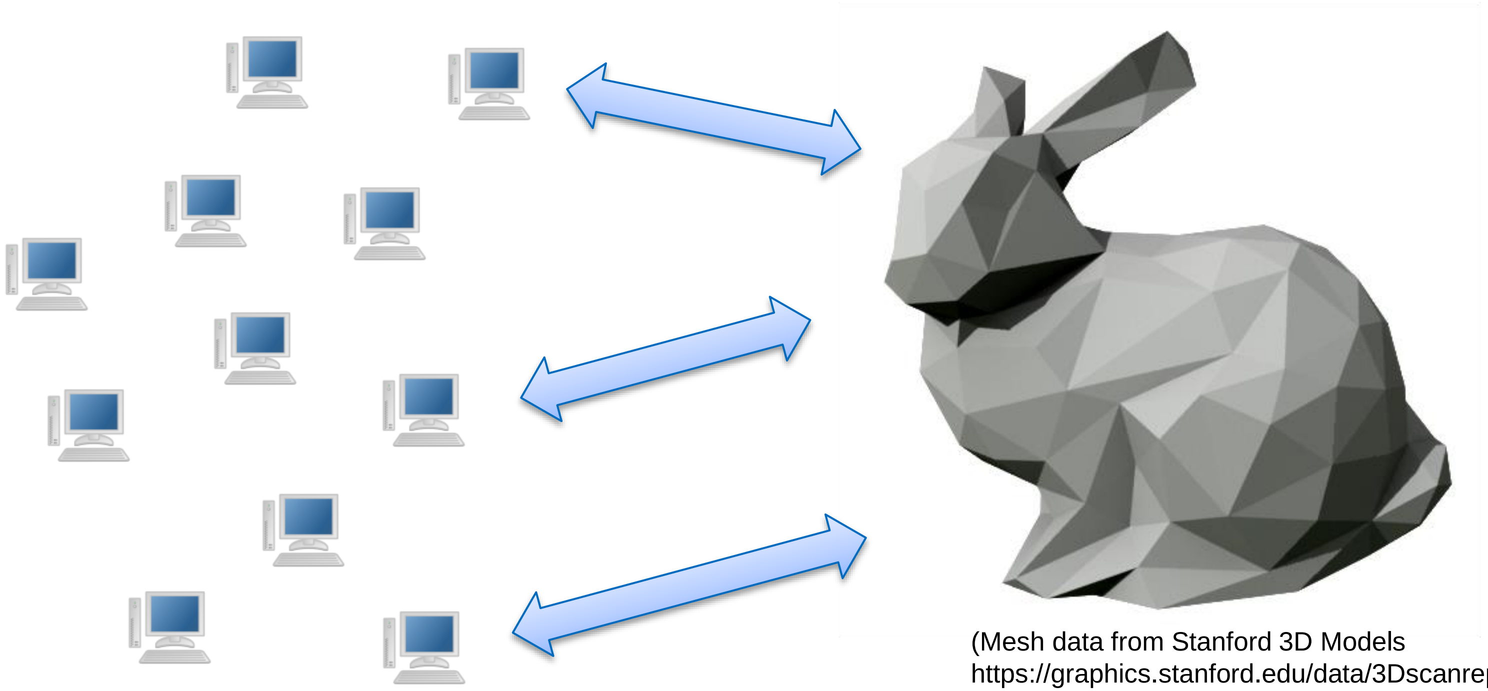
MMVE 2015, 20 March 2014, Portland

Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects

Concurrency Control for CVE

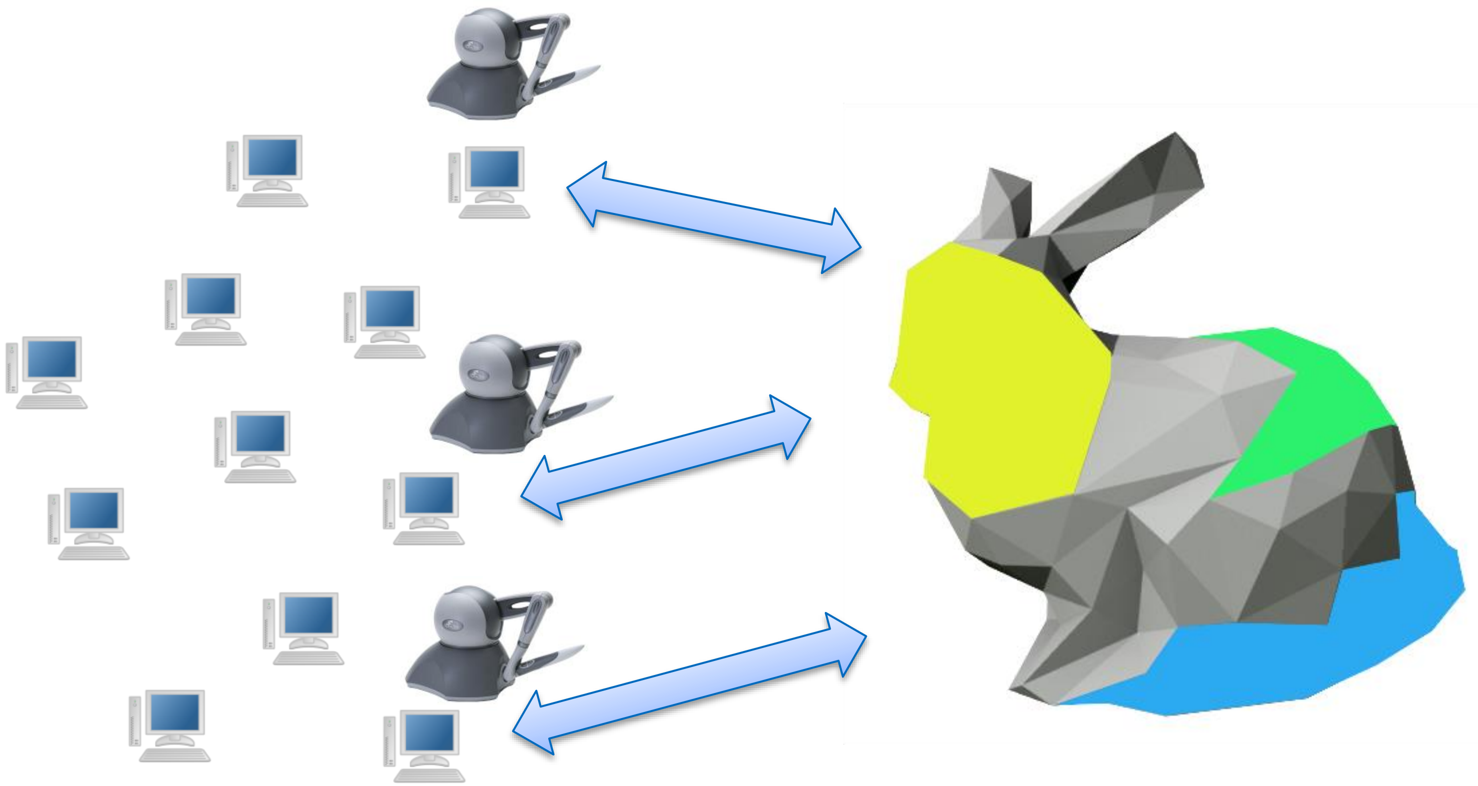
- Process of managing simultaneous execution of user transactions on shared virtual objects



(Mesh data from Stanford 3D Models https://graphics.stanford.edu/data/3Dscanrep/)

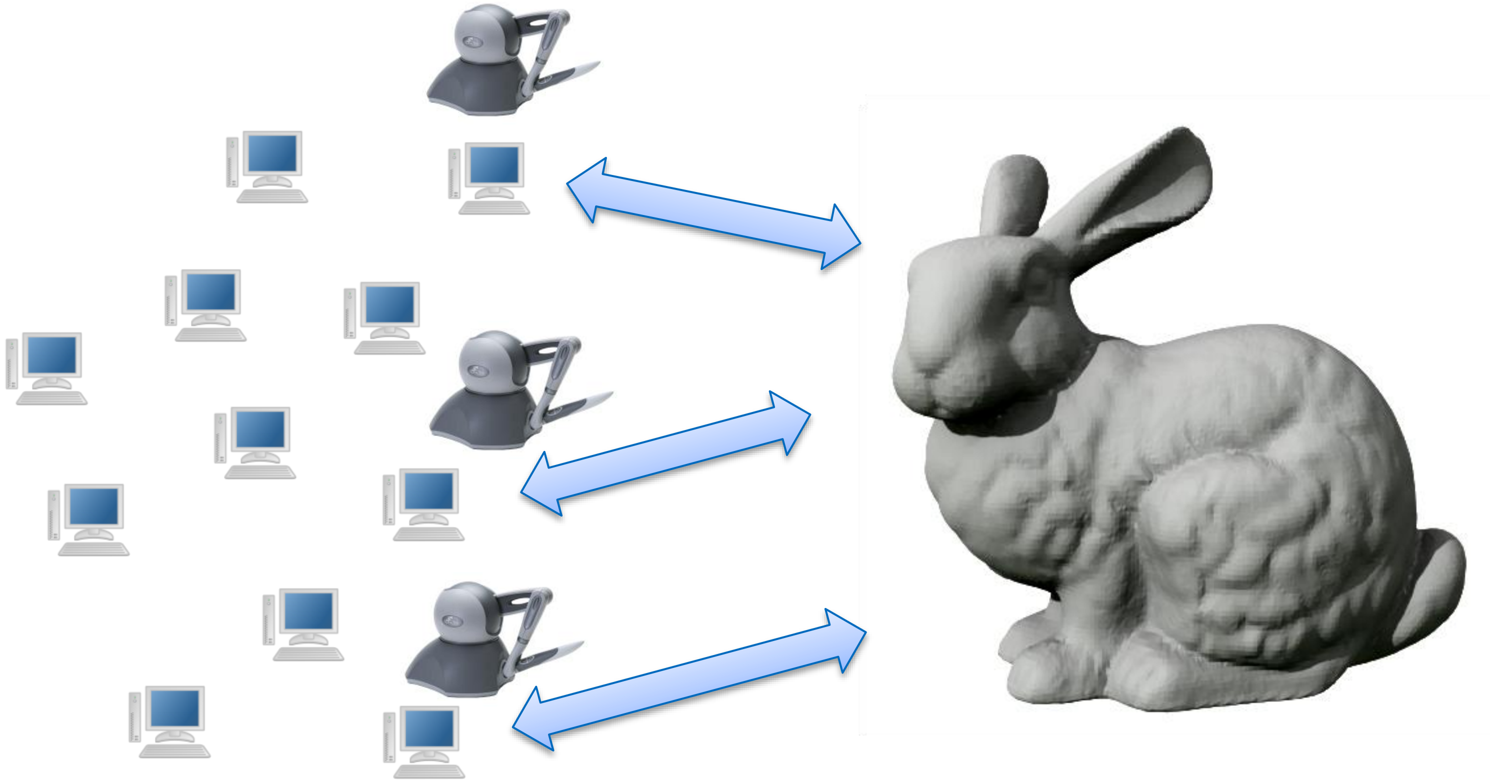
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects



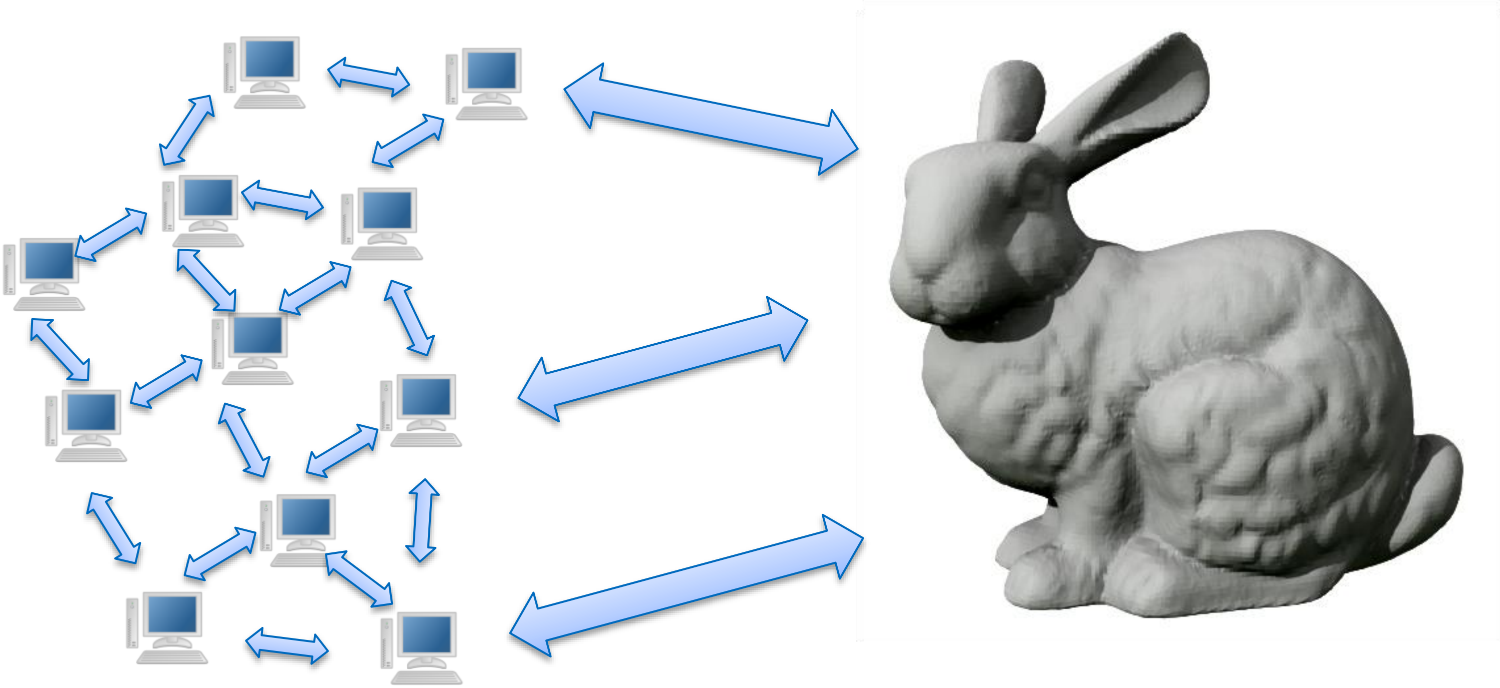
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects



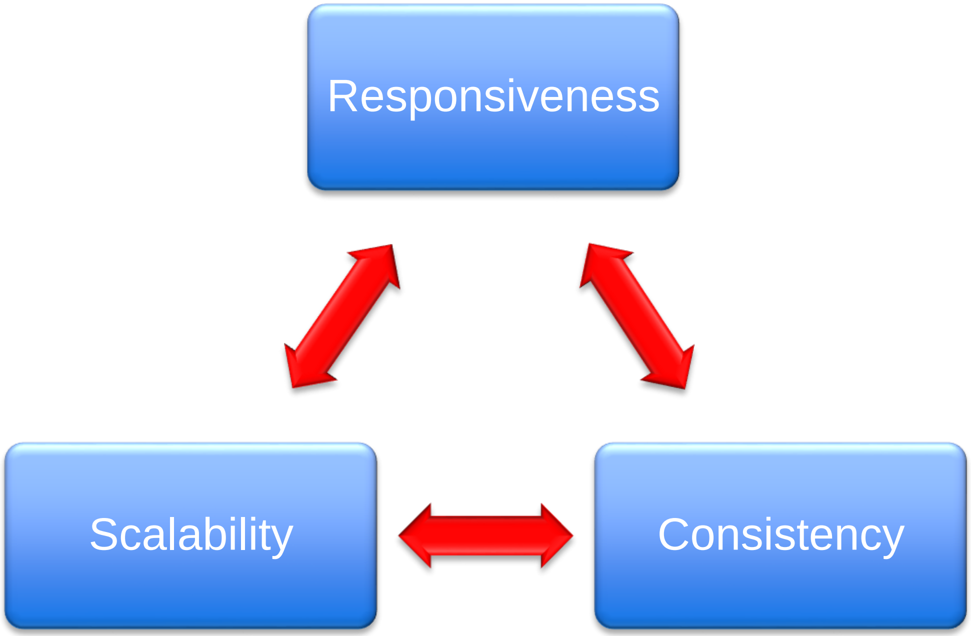
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects



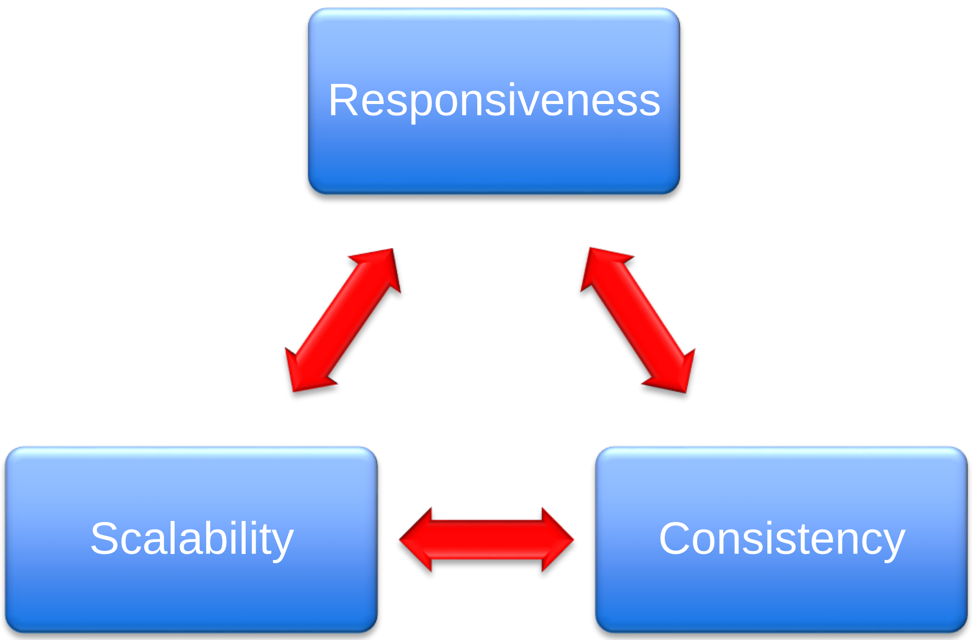
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects



Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects

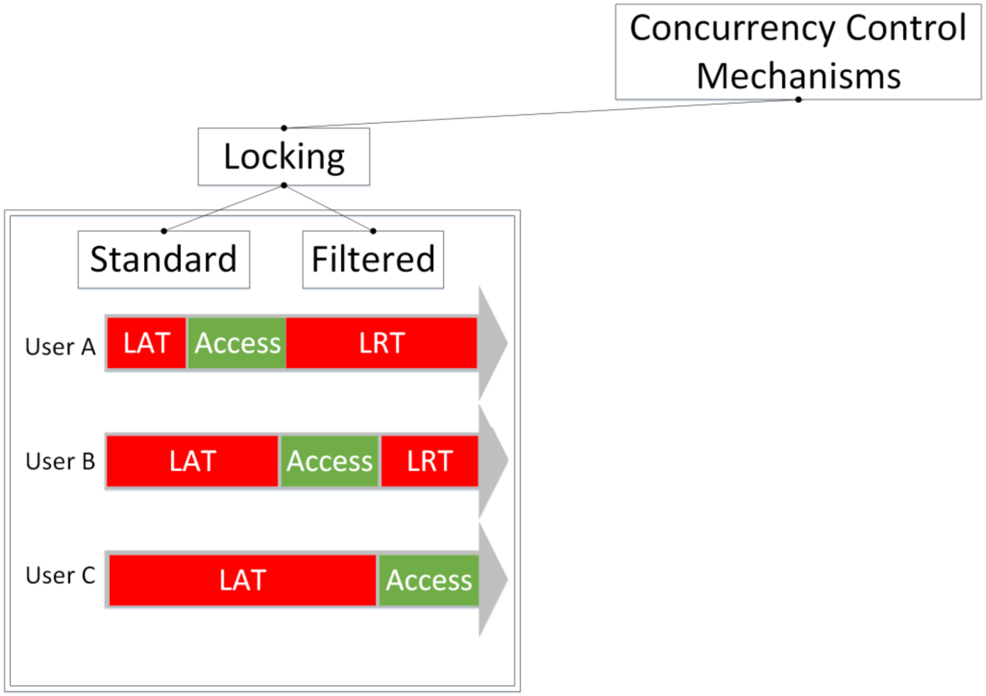


- Can lead to frustrated user experience or even user completely losing interest in the application [Roberts‘04][Bouckerche‘05]

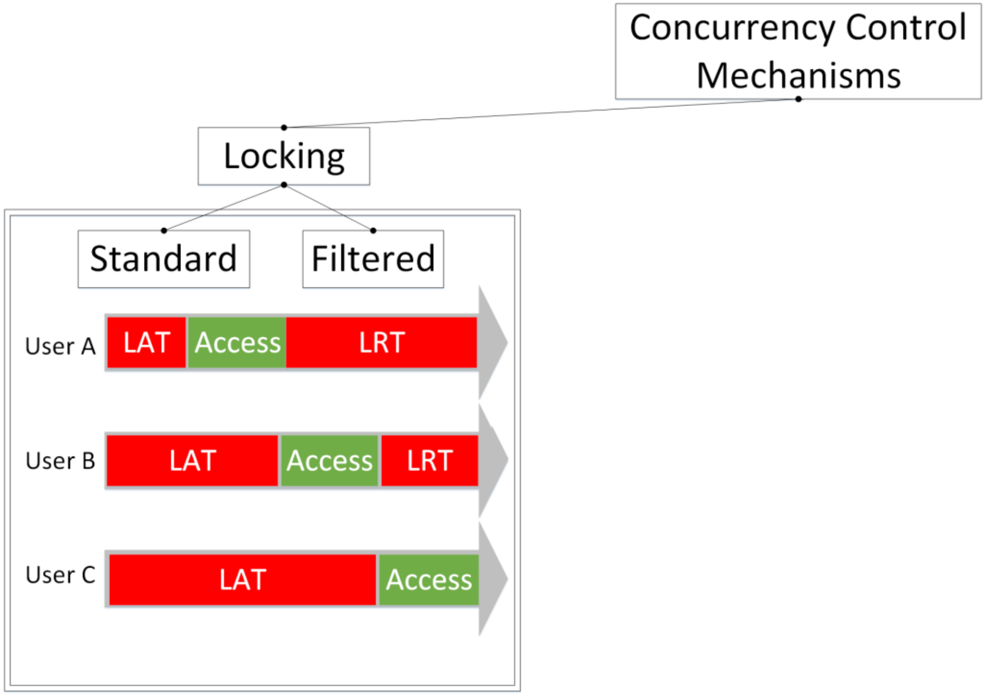
CCM for CVEs so far

Concurrency Control Mechanisms

CCM for CVEs so far

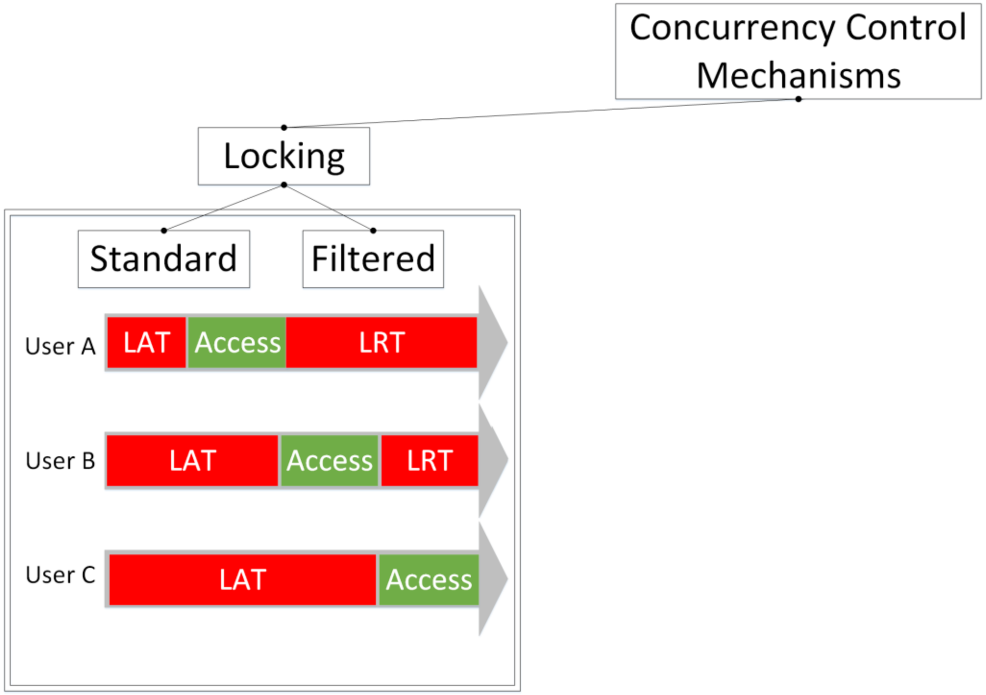


CCM for CVEs so far



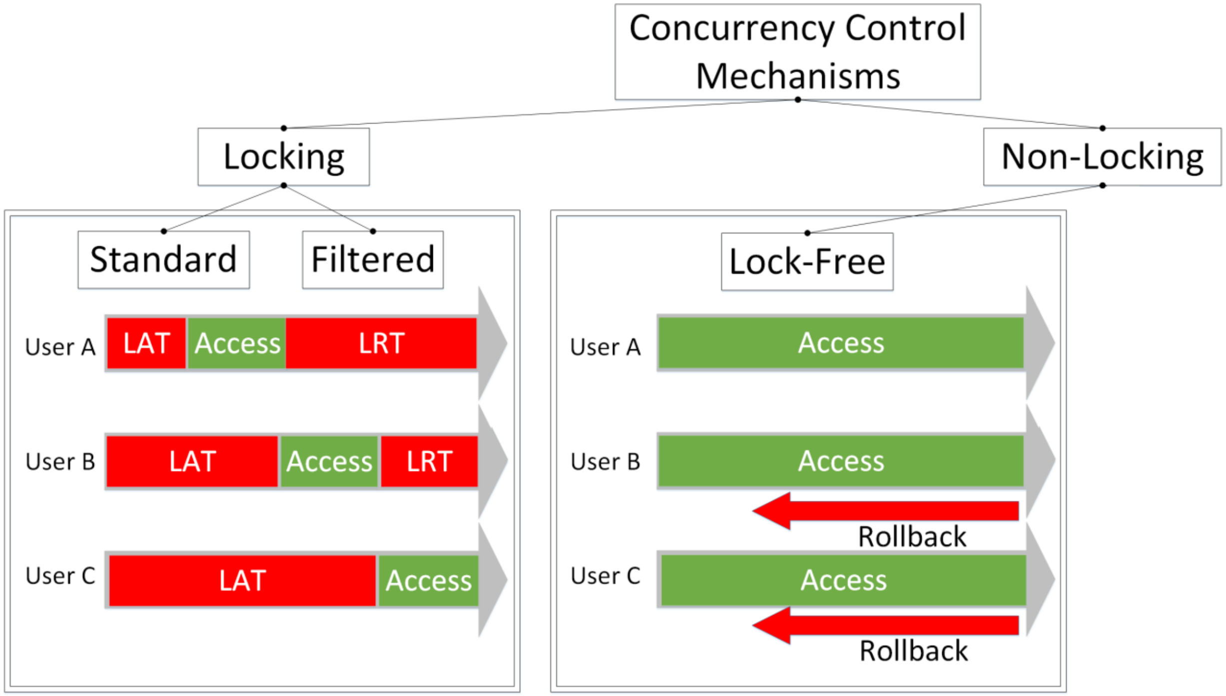
- VSculpt: A distributed virtual environment for collaborative design [Li‘03]
- Architectures for shared haptic virtual environments [Buttolo‘97]

CCM for CVEs so far



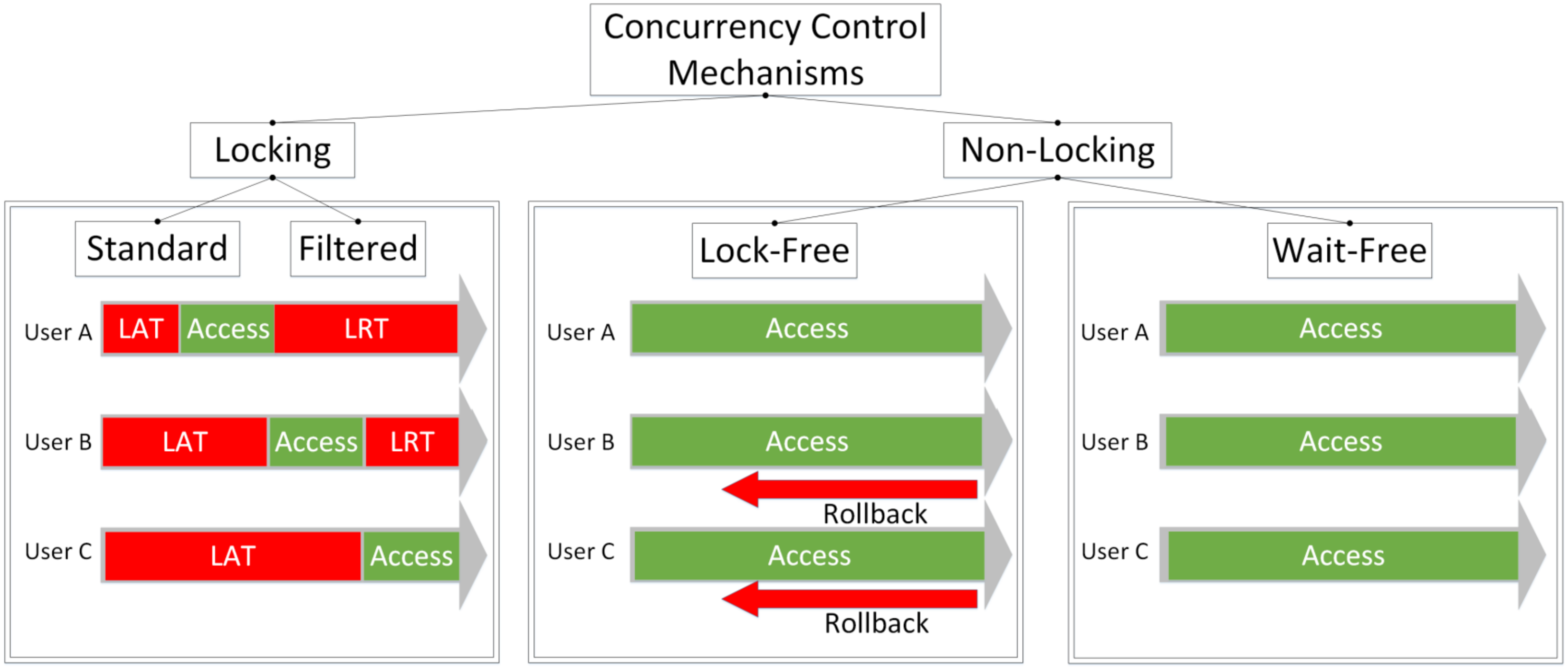
- ATLAS – A scalable network framework for distributed virtual environments [Lee‘07],
- Scalable prediction based concurrency control for distributed virtual environments [Yang‘00]

CCM for CVEs so far



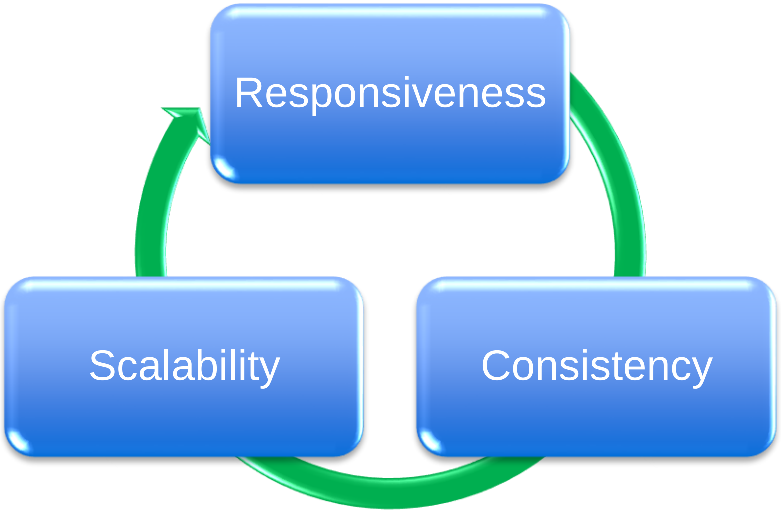
- Performance evaluation of compromised synchronization control mechanism for distributed virtual environment [Wongwirat'06]

CCM for CVEs so far



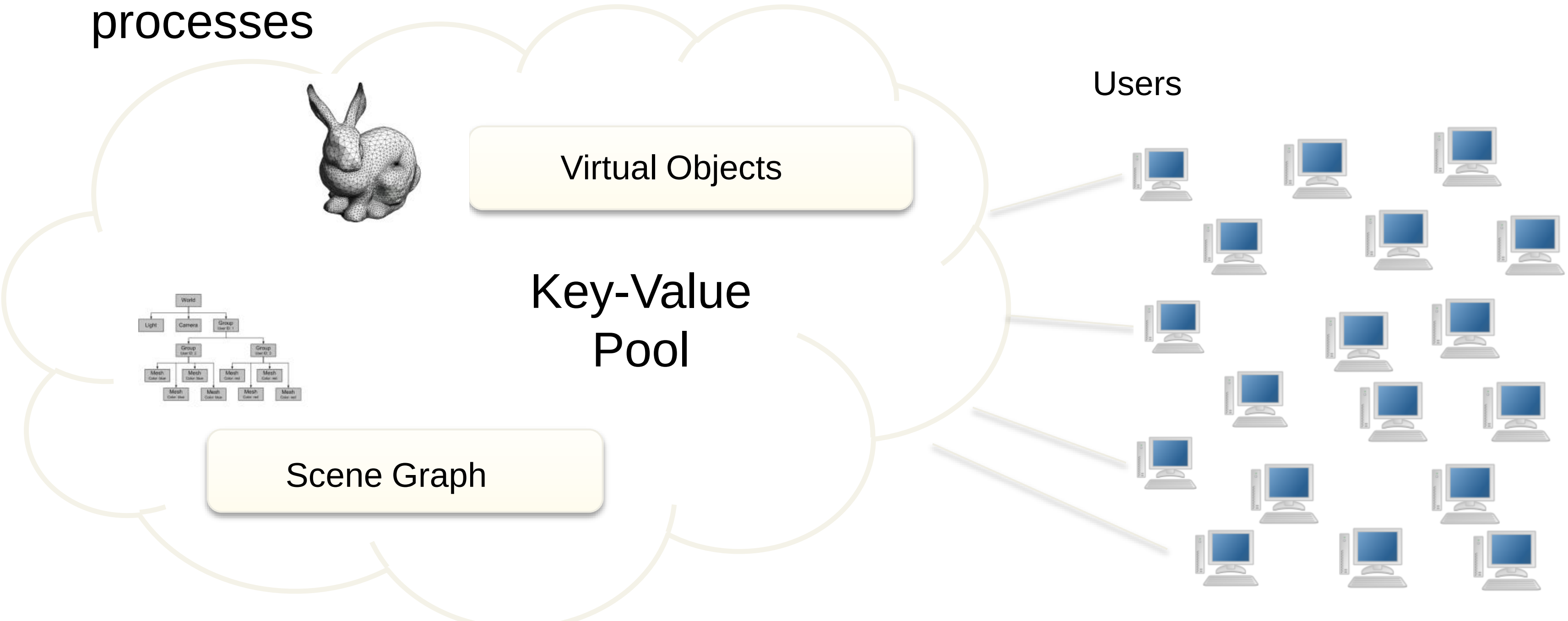
Our Contribution

- Novel approach to concurrency control for massively collaborative virtual environments
  - Not affected by network delays
  - No problems from previous approaches like deadlocks or starvation
- High performance access
  - Almost constant runtime with very low synchronisation overhead
  - Multiple wait-free read and write operations

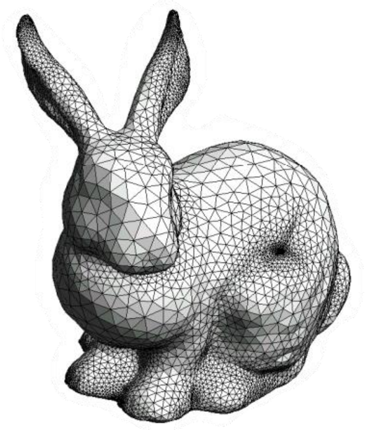


Basic Idea

- Assignment of unique key-value pair to each data packet which is exchanged between users and virtual objects
- Key-value pool holds complete shared world state
- De-coupling and parallelization of read, write and data deletion processes



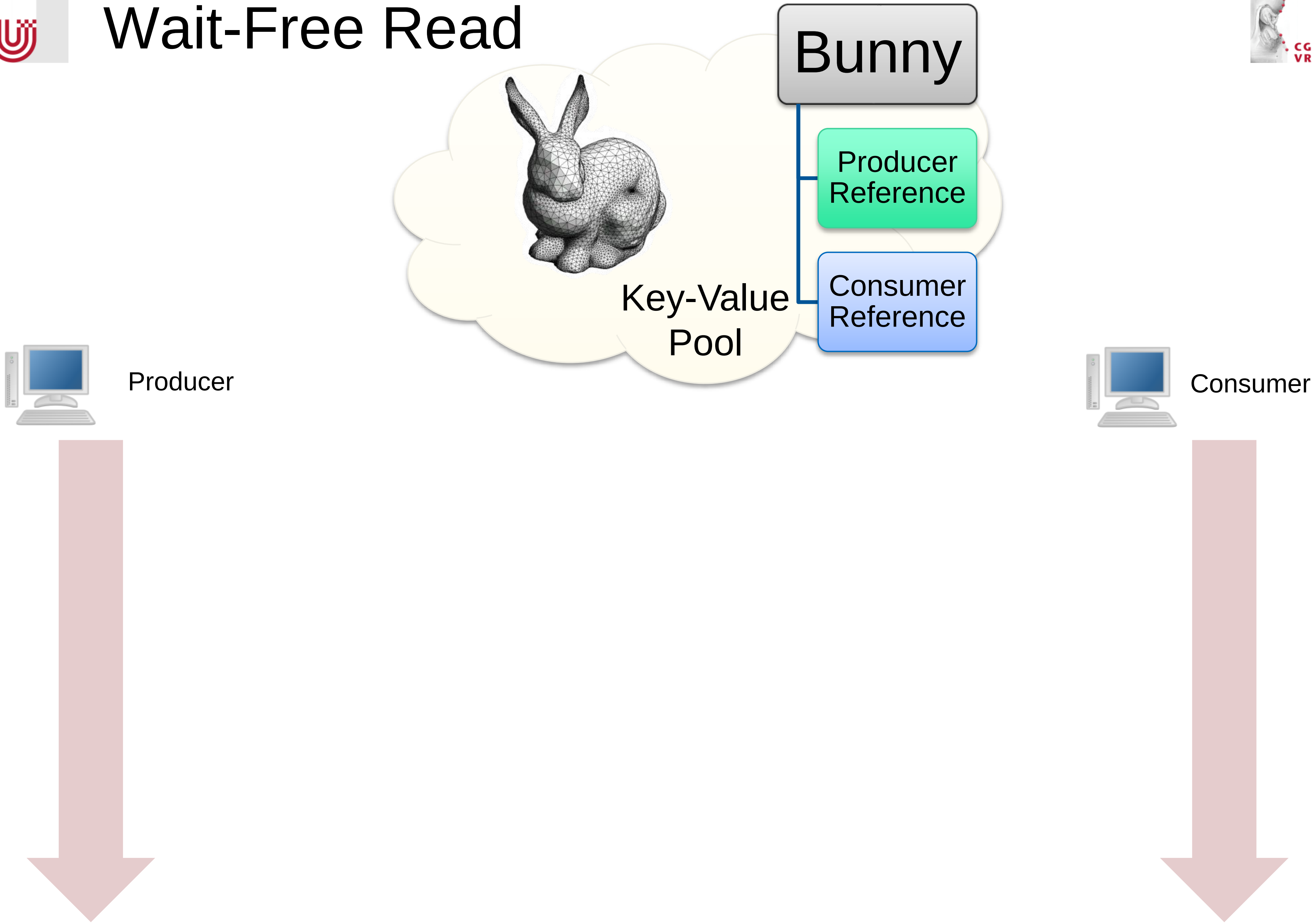
Wait-Free Read



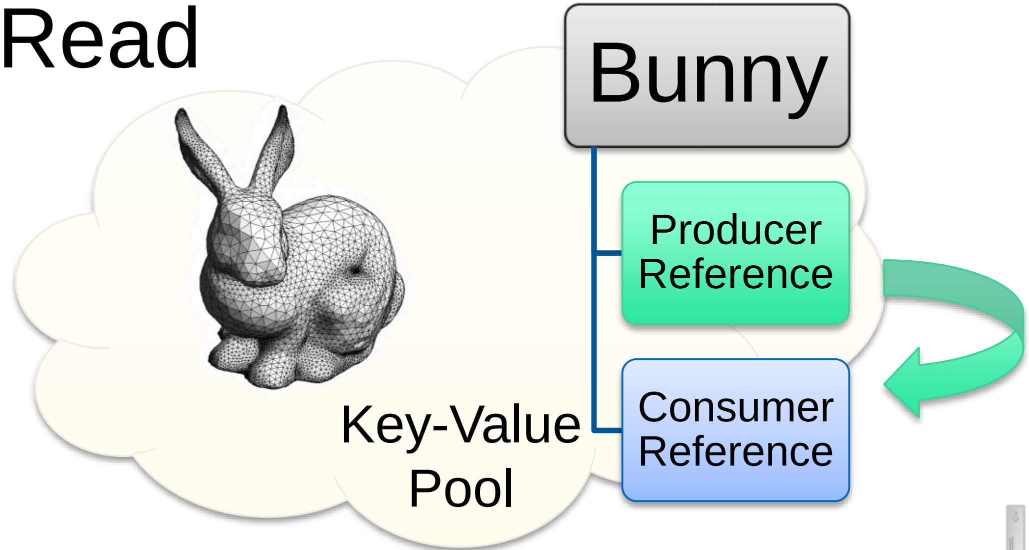
Producer

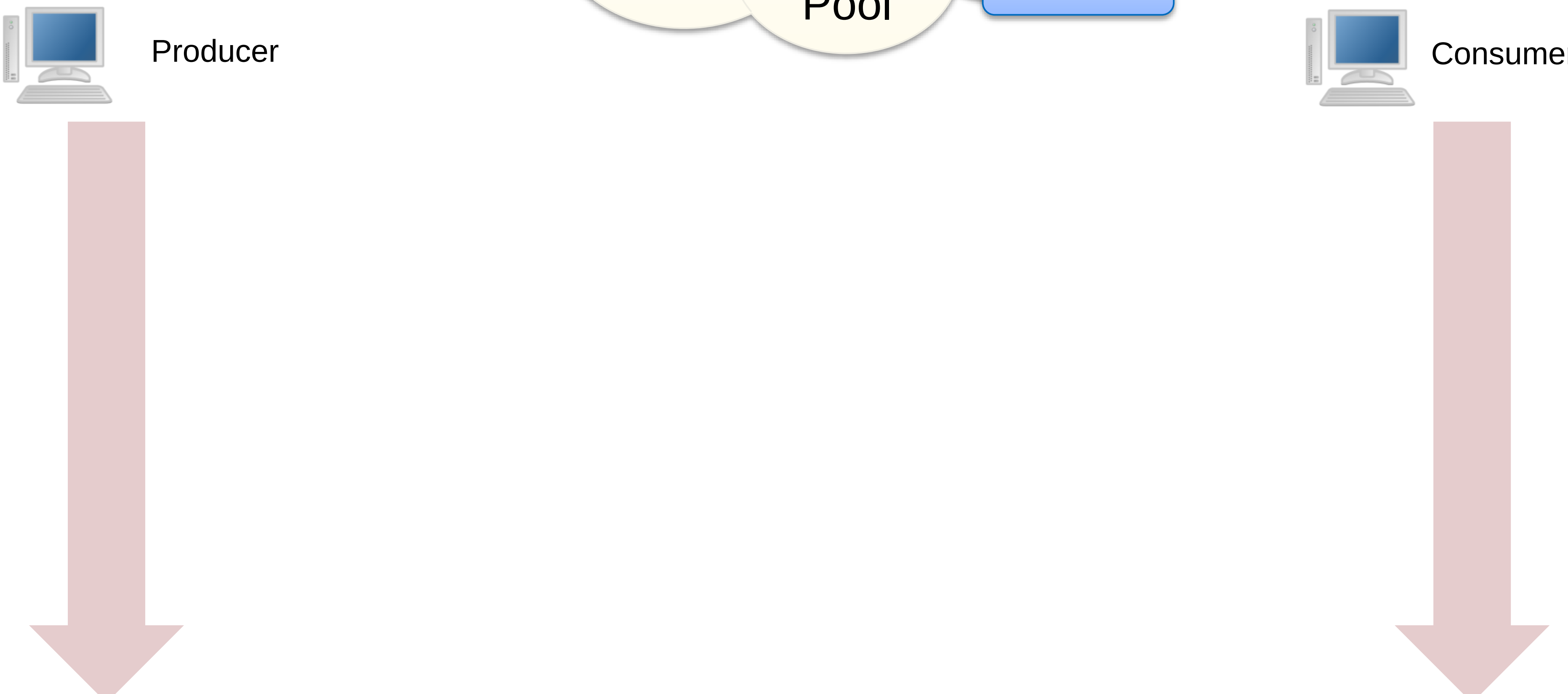
Consumer

Wait-Free Read

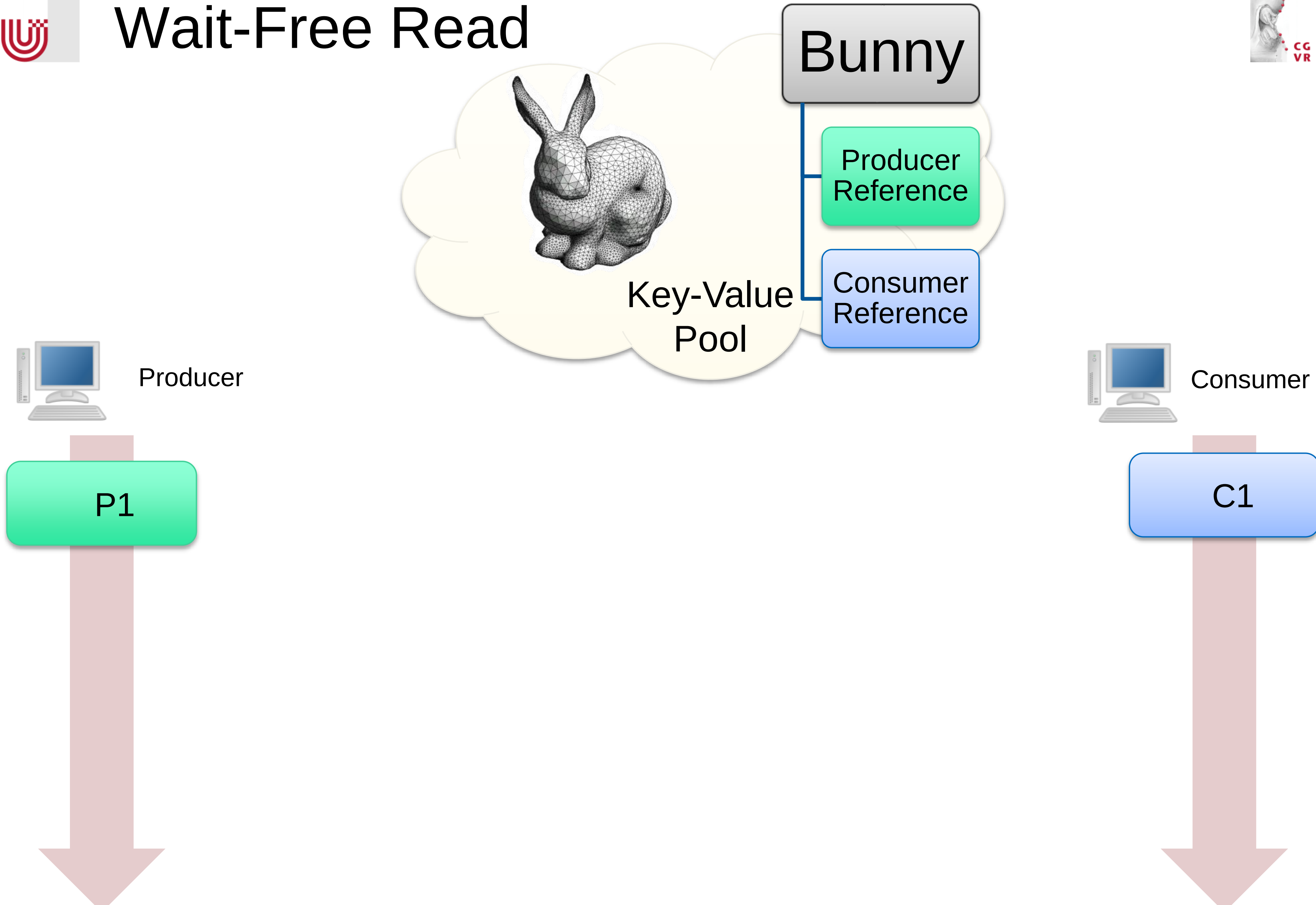


Wait-Free Read

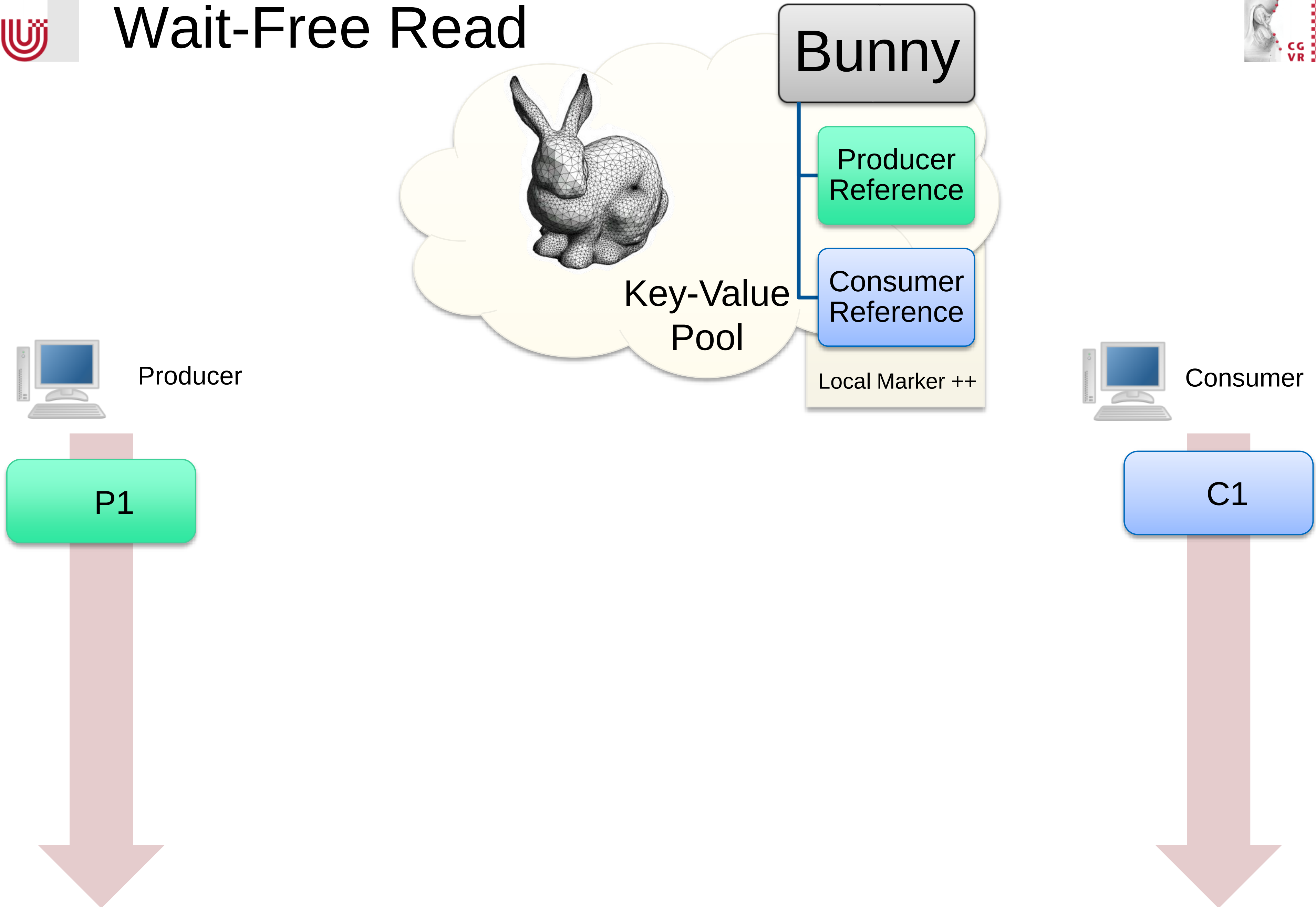




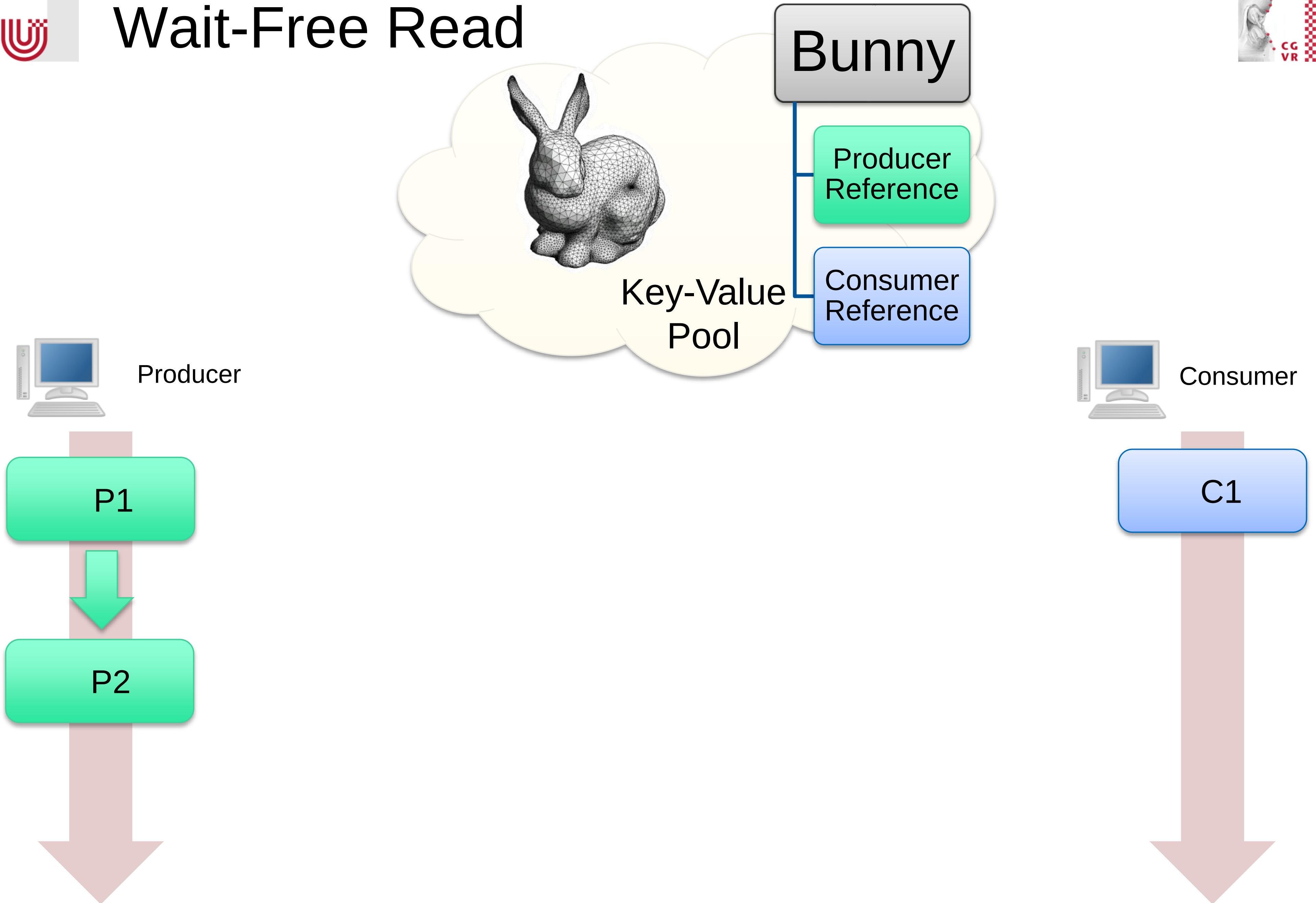
Wait-Free Read



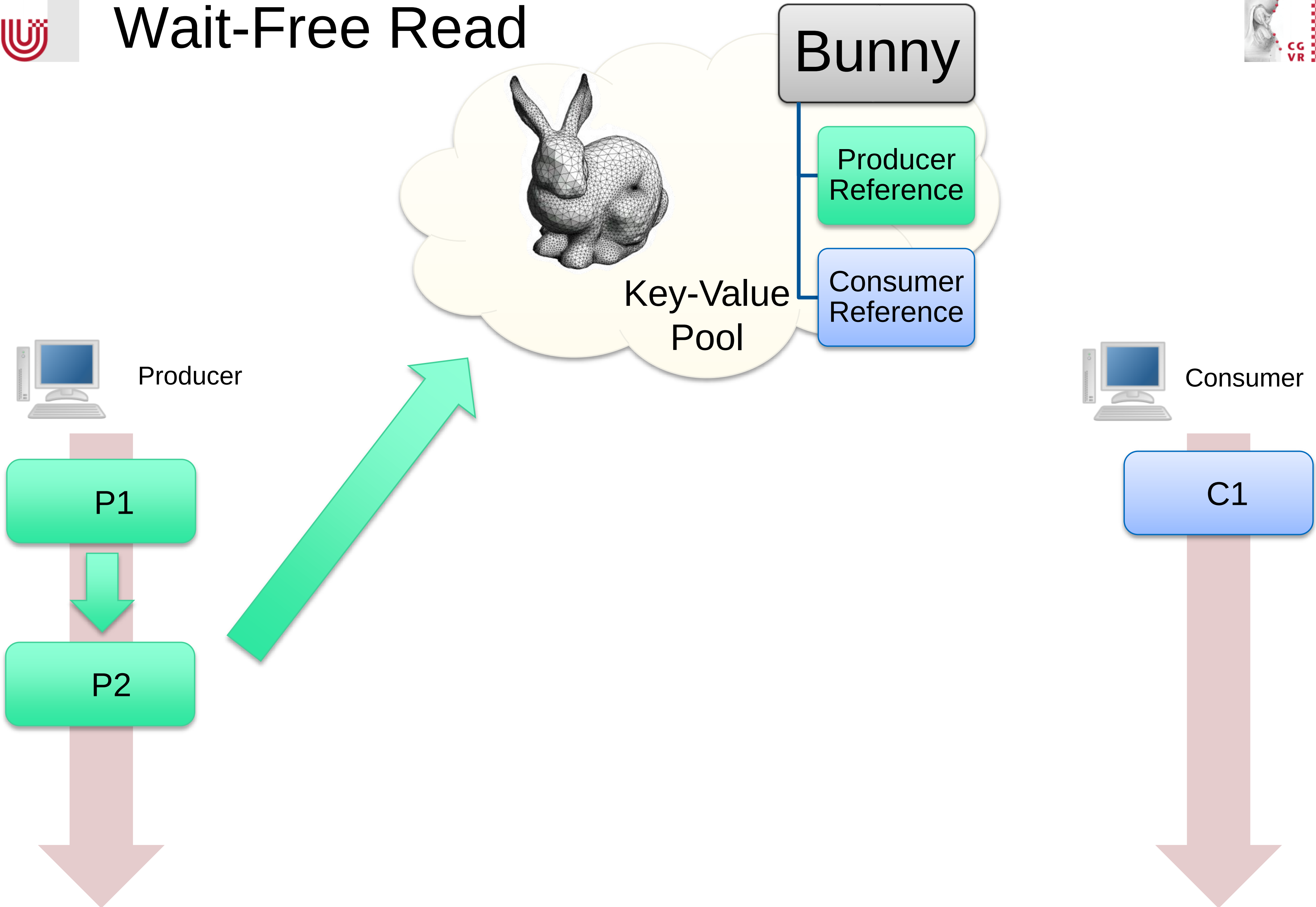
Wait-Free Read



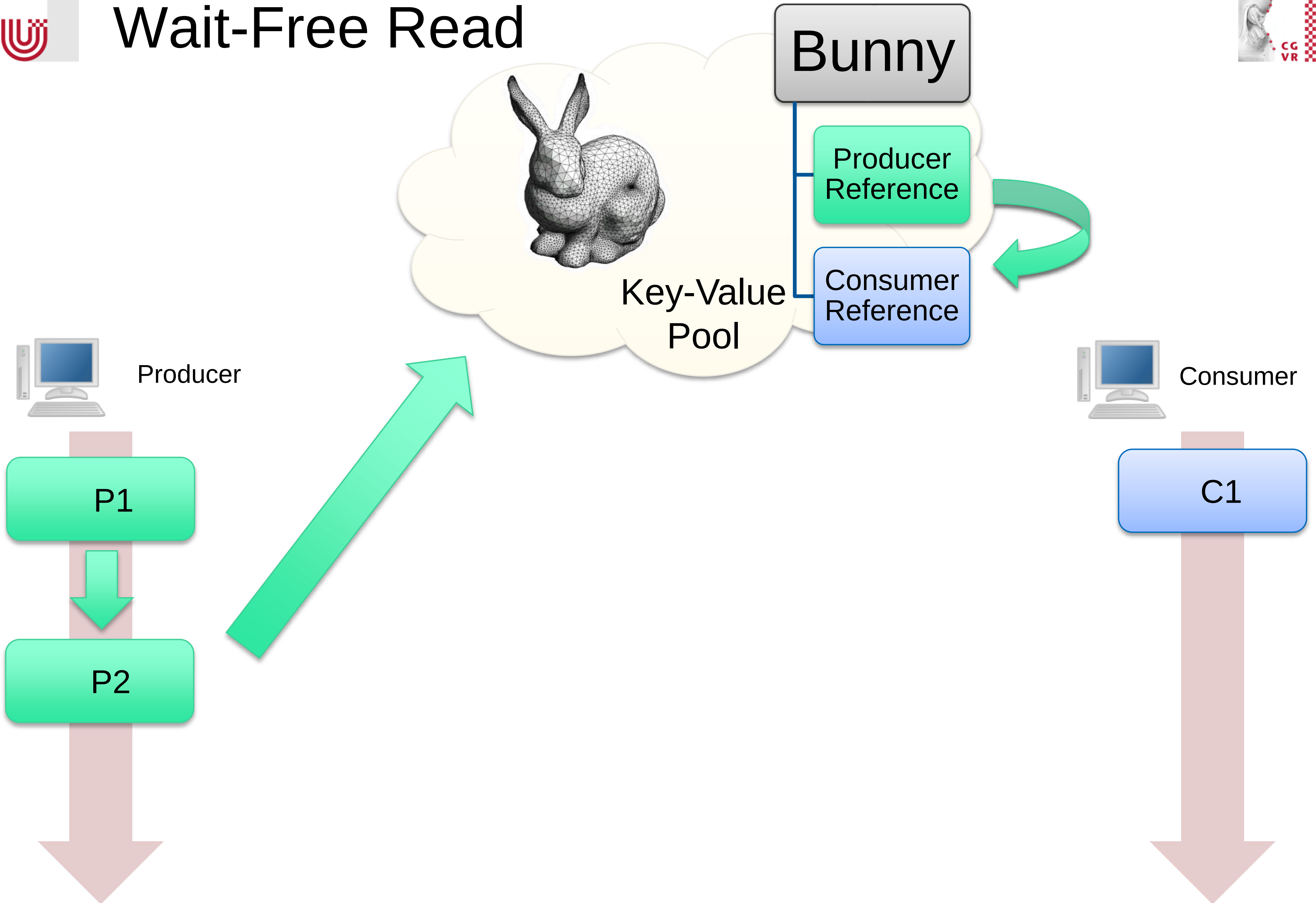
Wait-Free Read



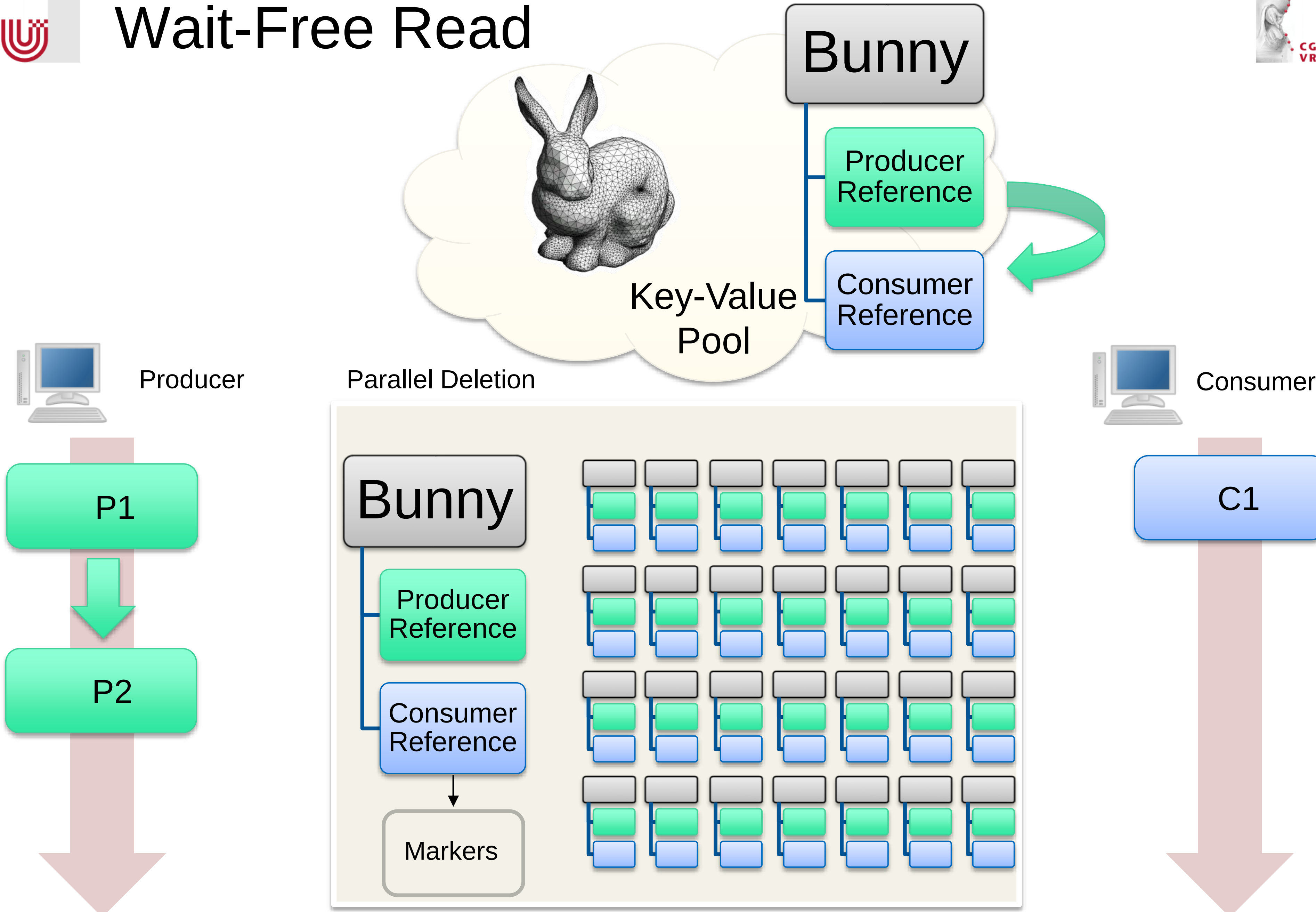
Wait-Free Read



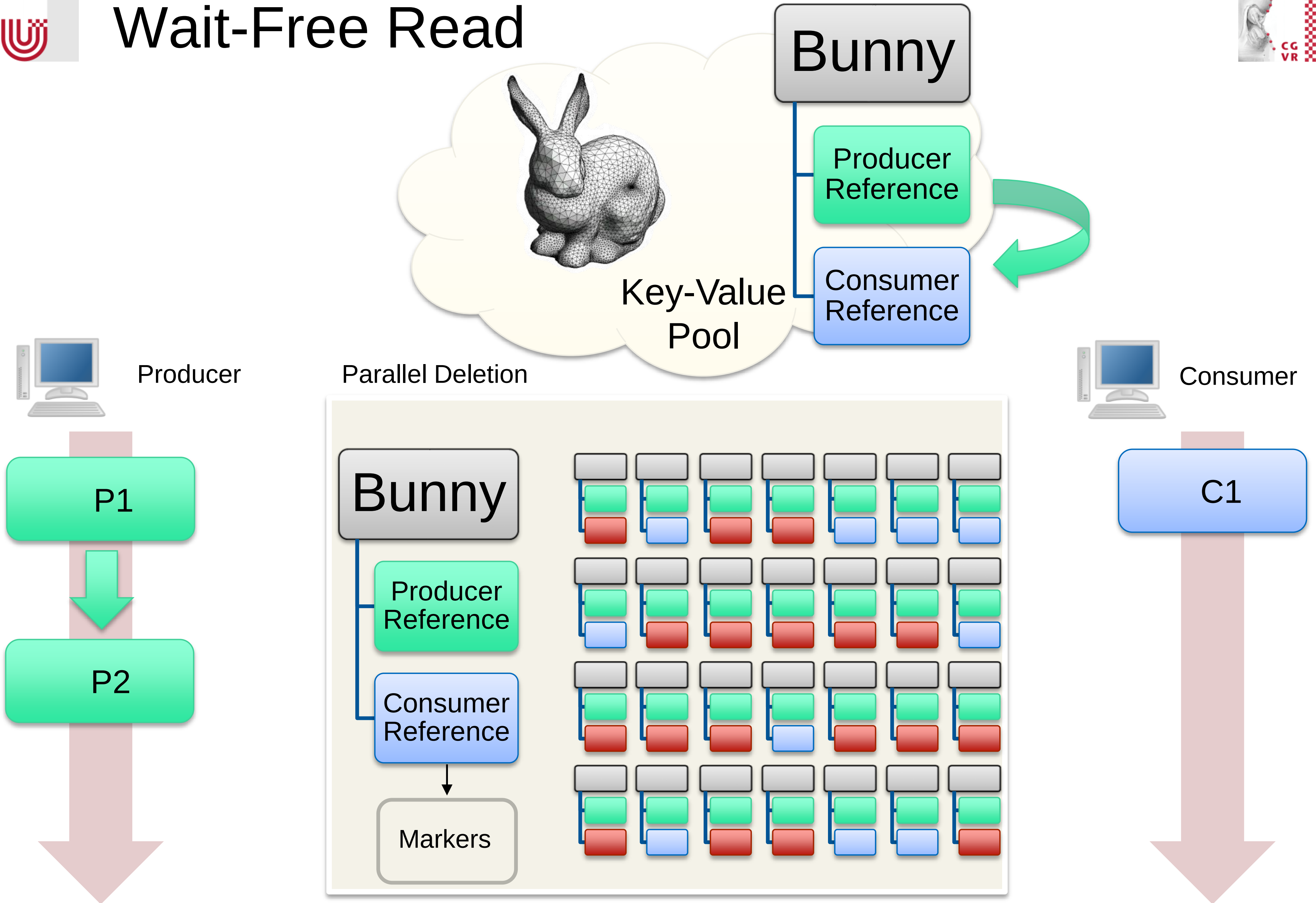
Wait-Free Read



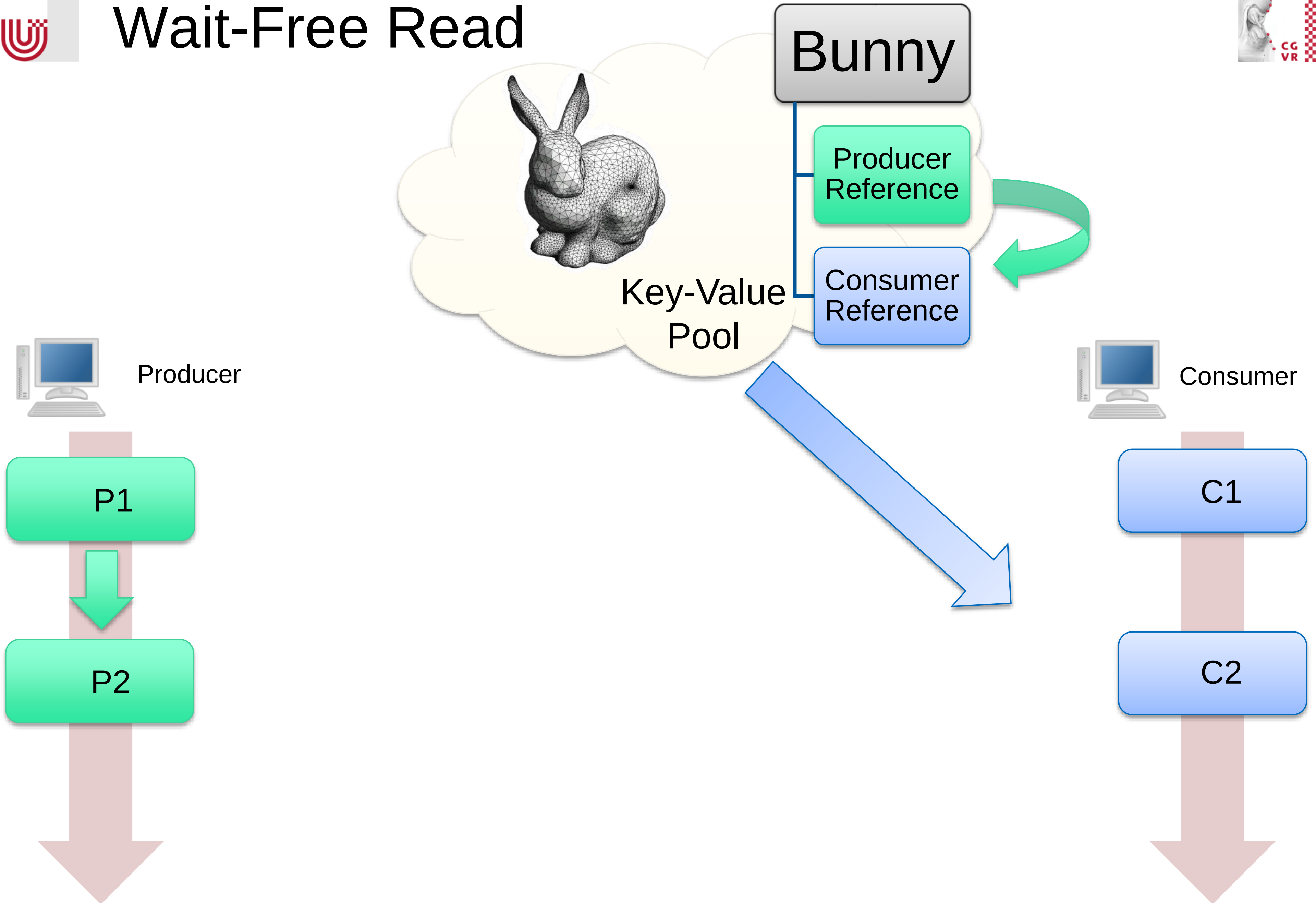
Wait-Free Read



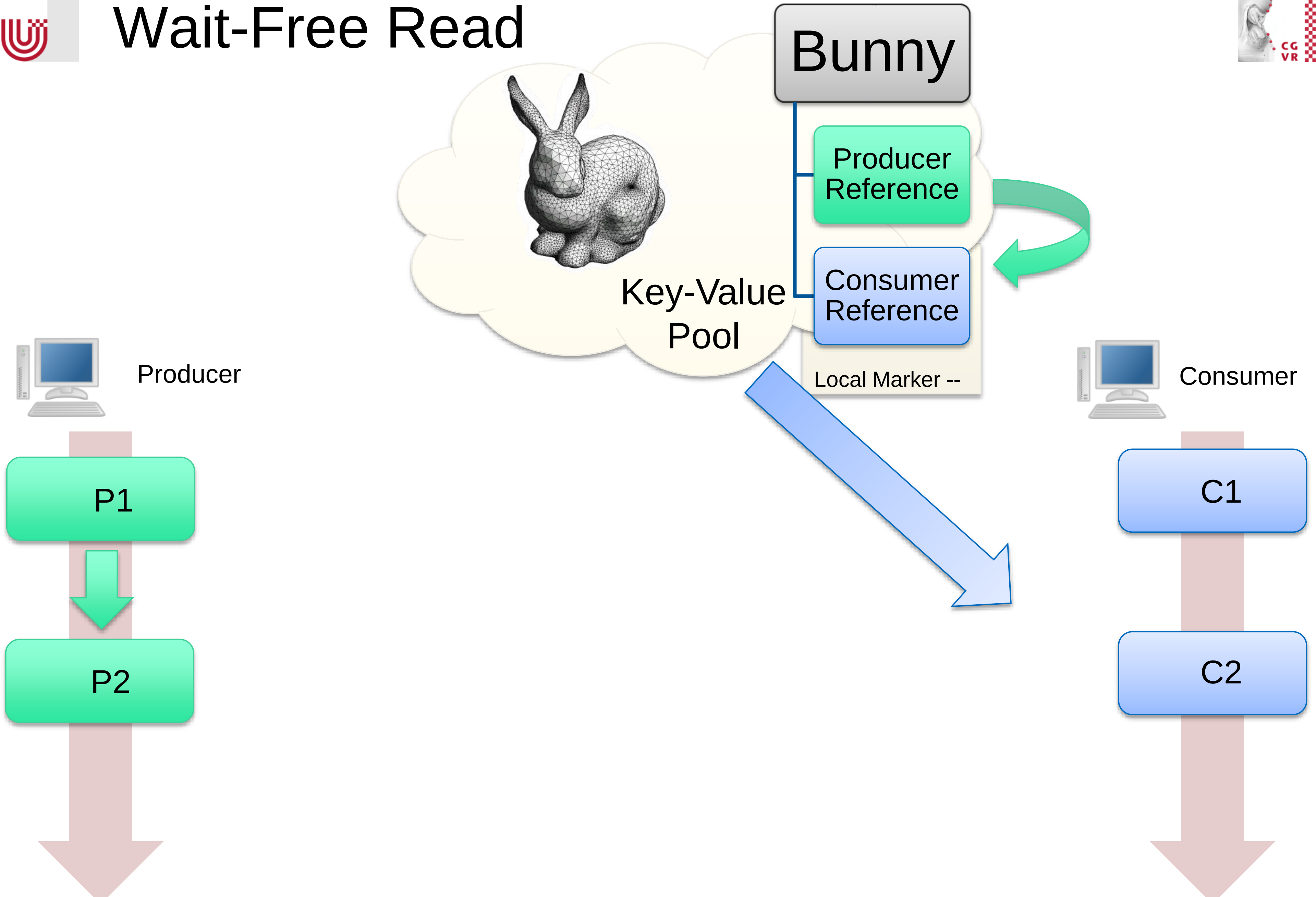
Wait-Free Read



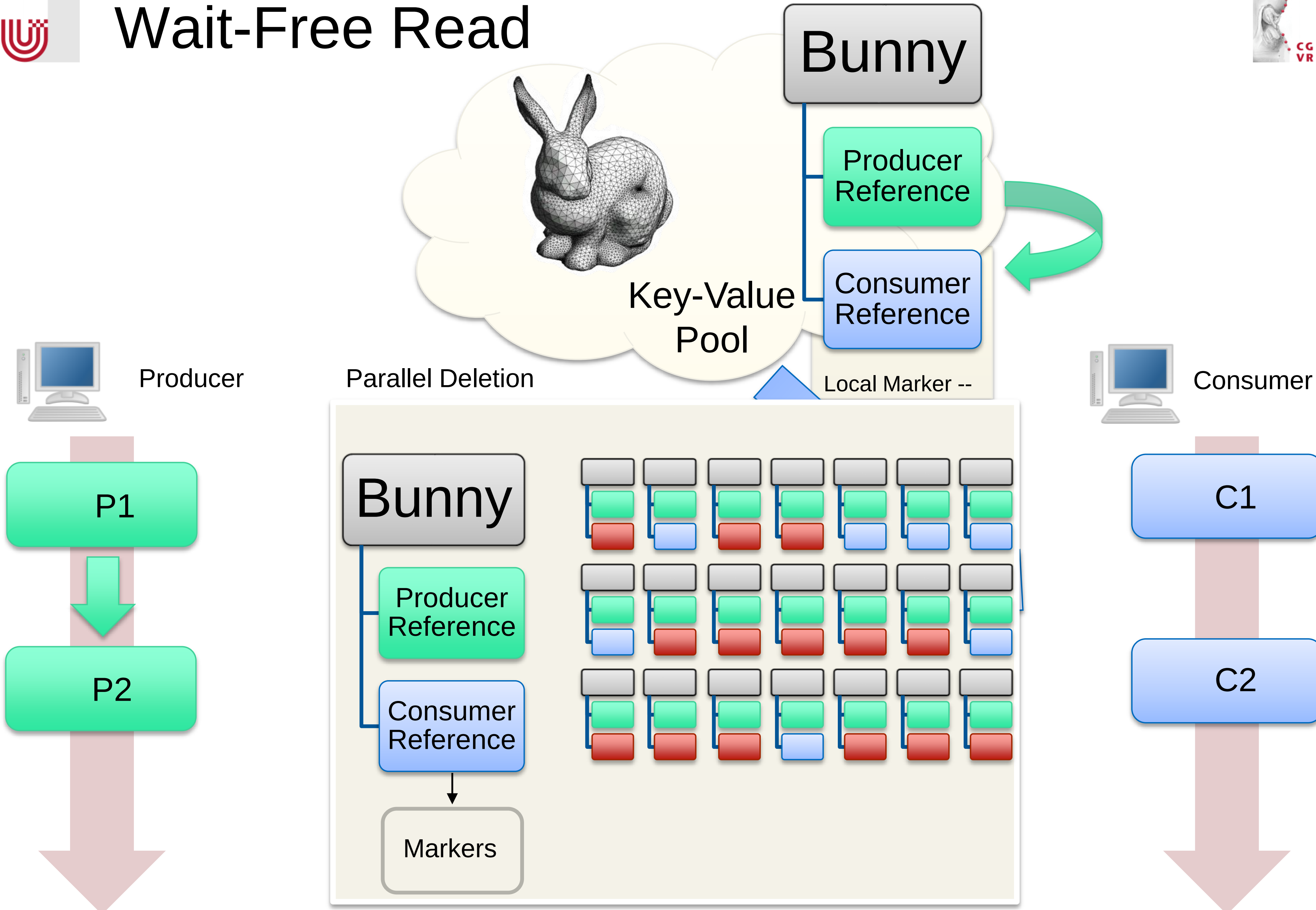
Wait-Free Read



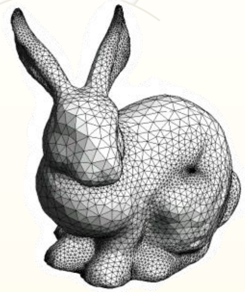
Wait-Free Read



Wait-Free Read



Wait-Free Write



Bunny

Producer Reference

Consumer Reference

Key-Value Pool

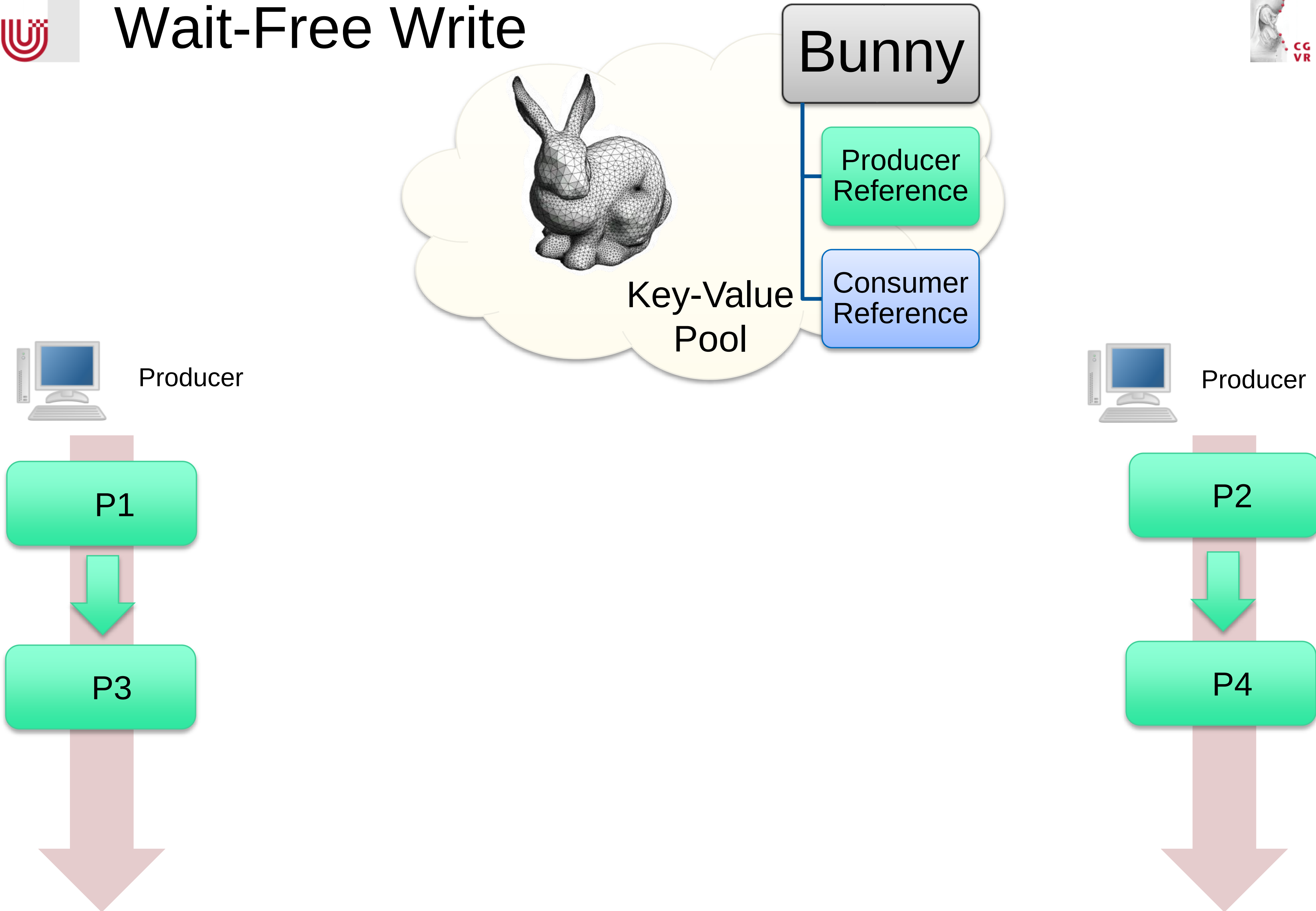
Producer

Producer

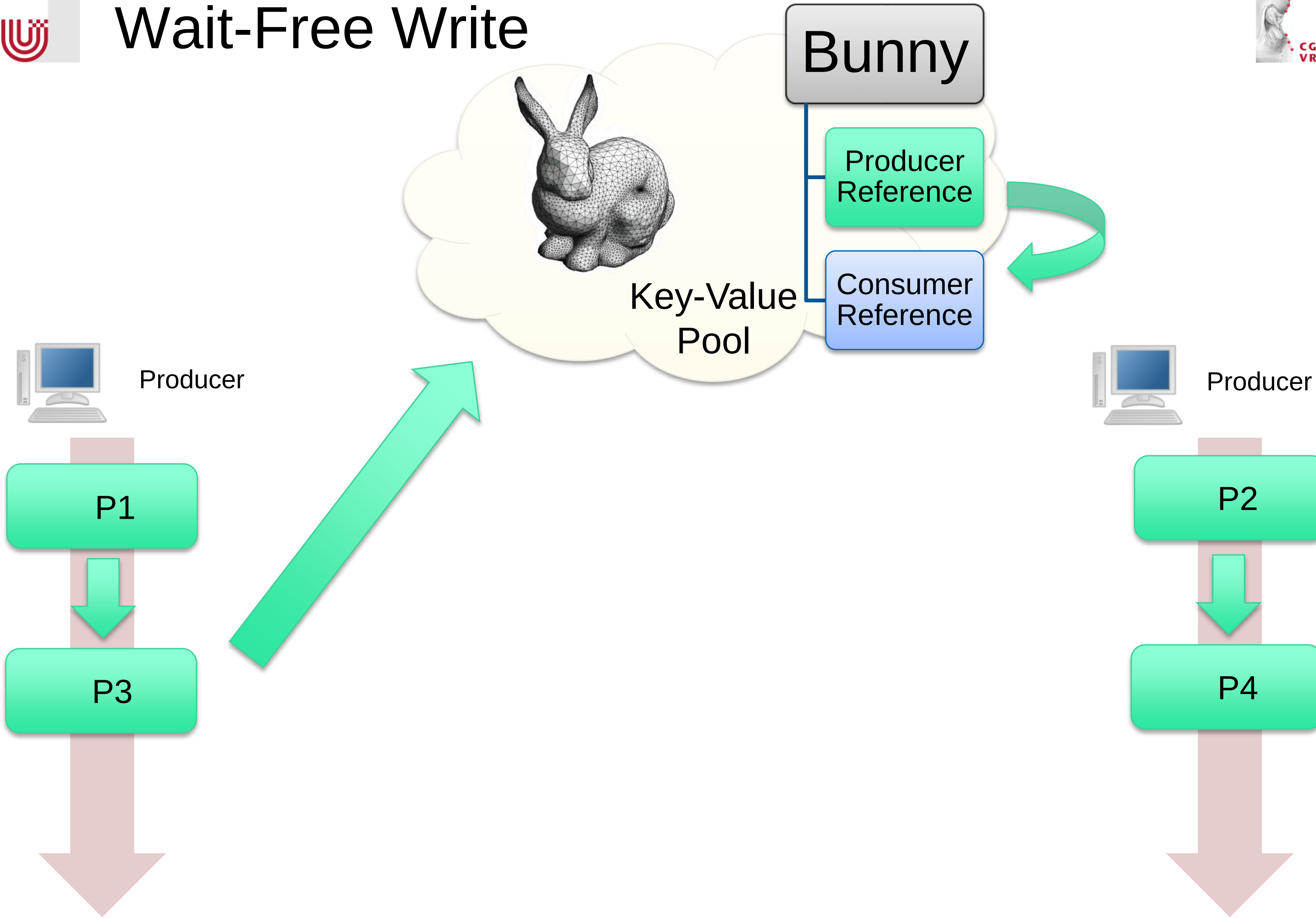
P1

P2

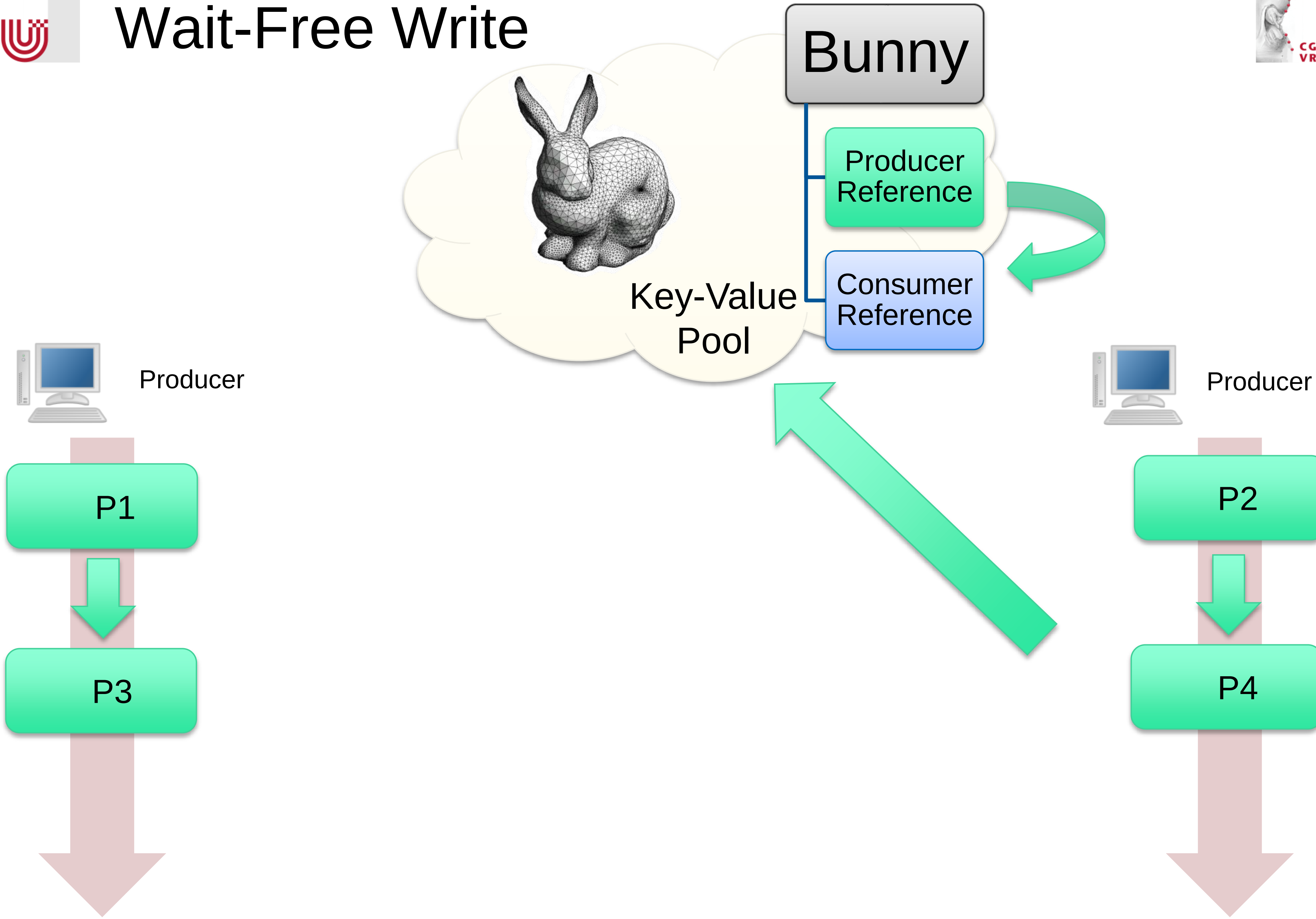
Wait-Free Write



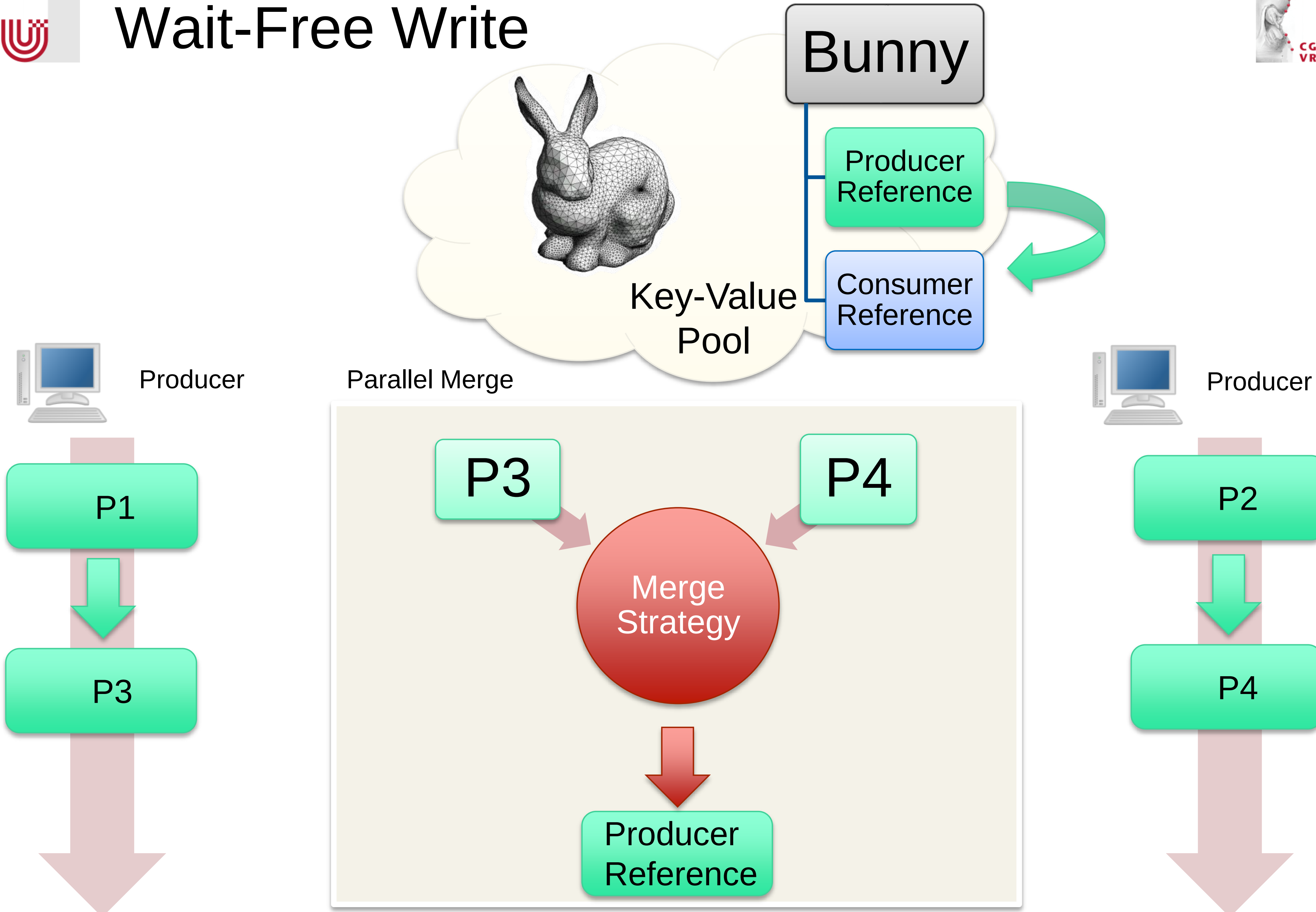
Wait-Free Write



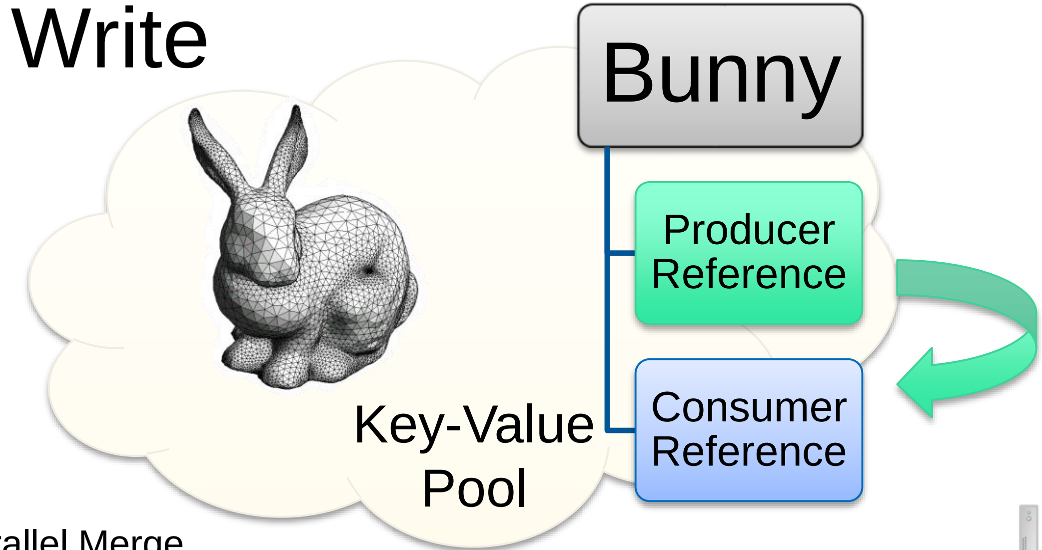
Wait-Free Write

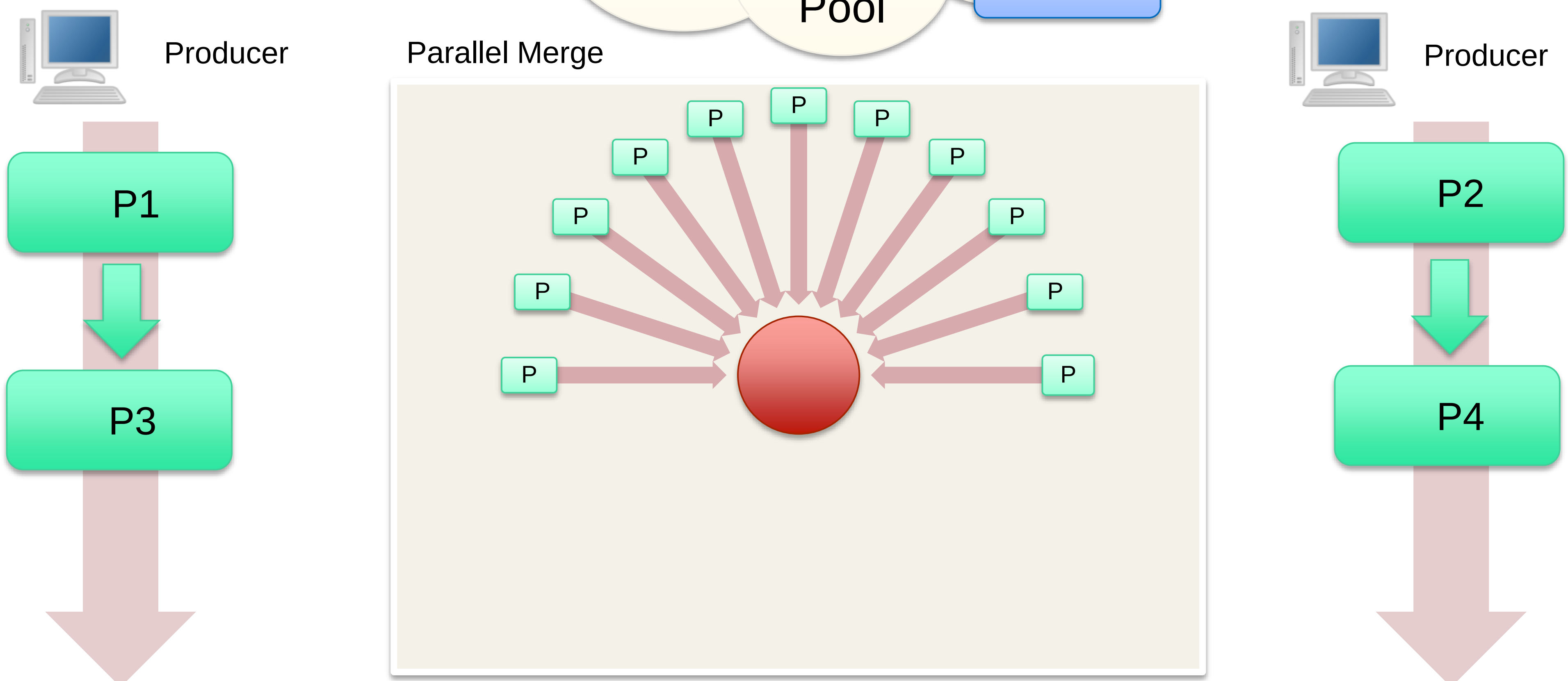


Wait-Free Write

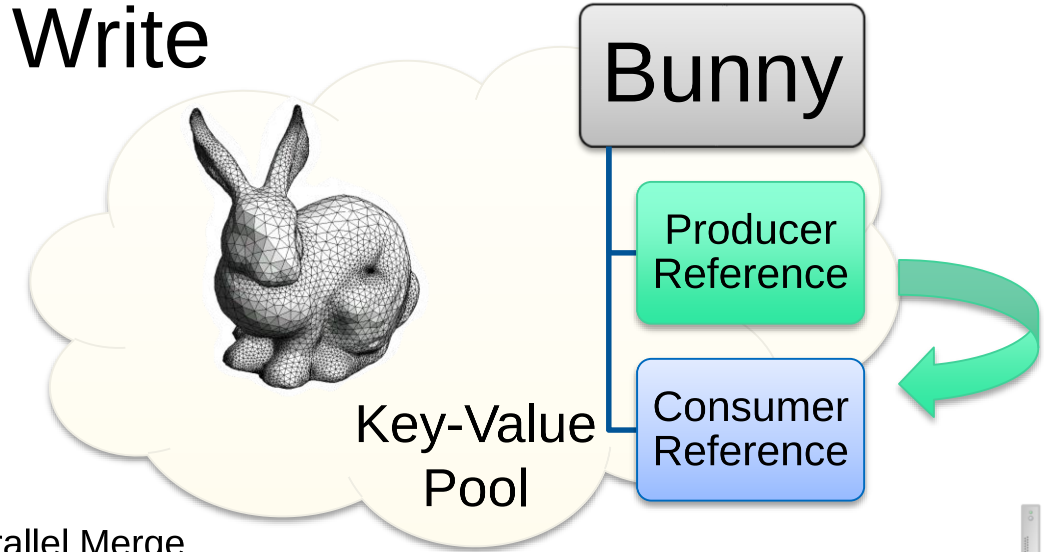


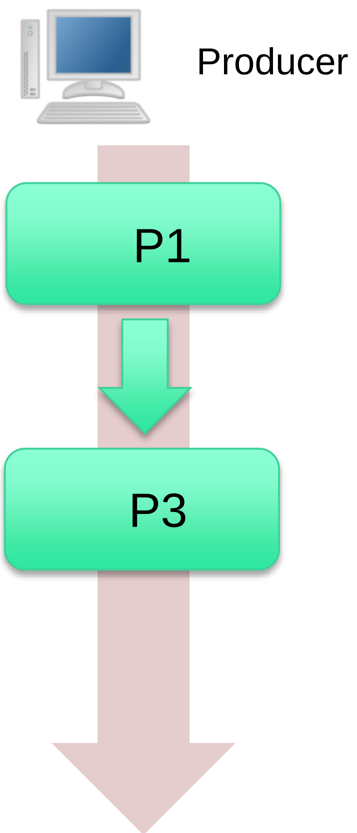
Wait-Free Write



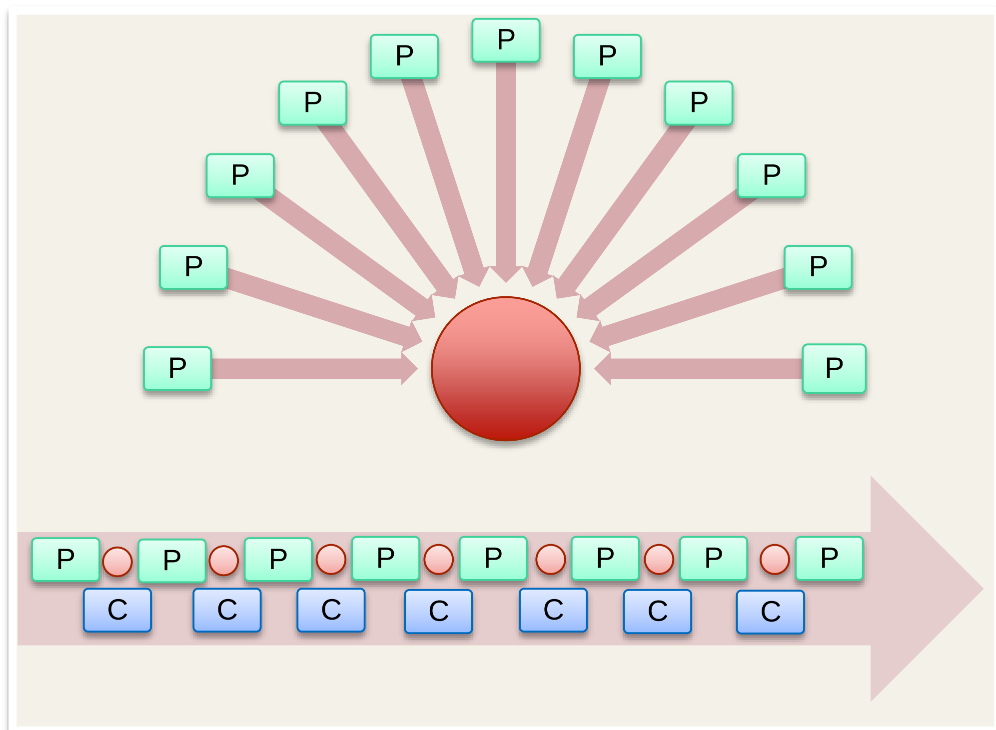


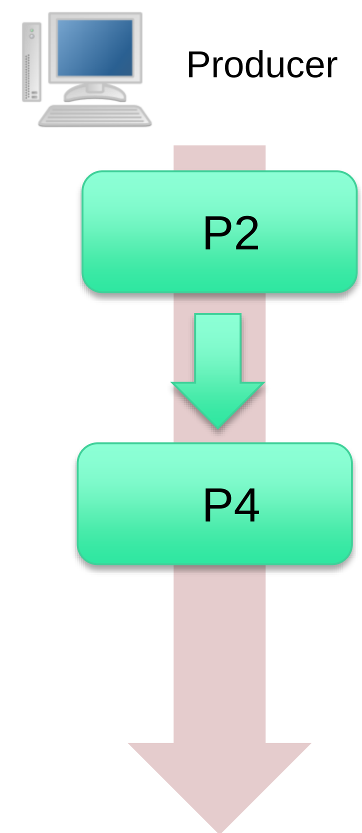
Wait-Free Write



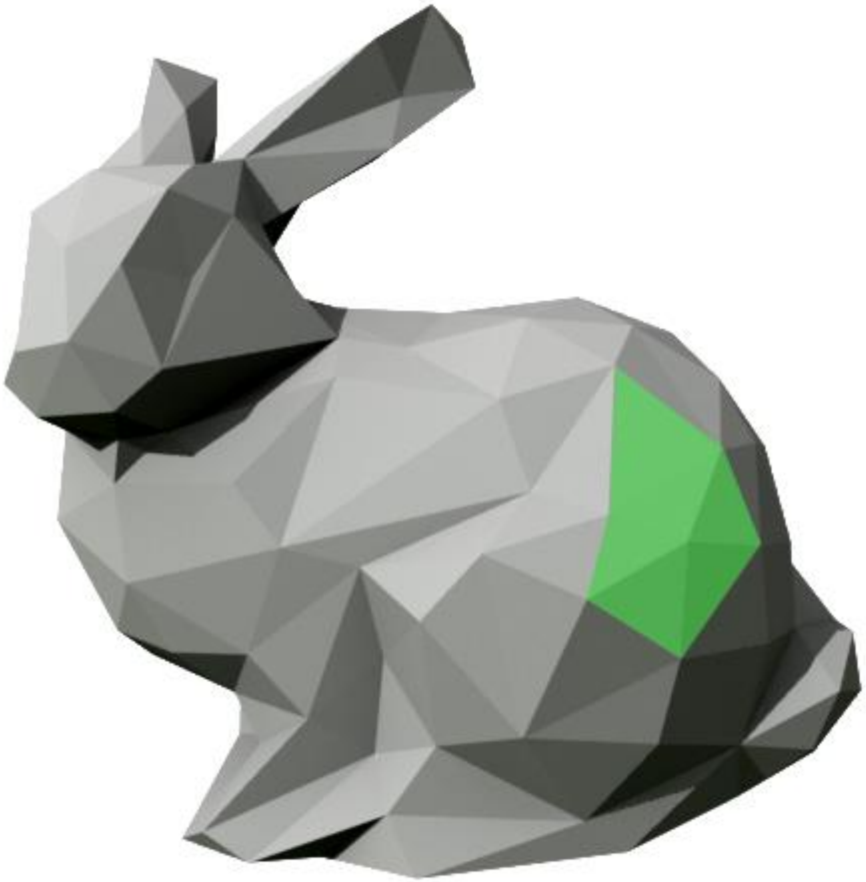


Parallel Merge





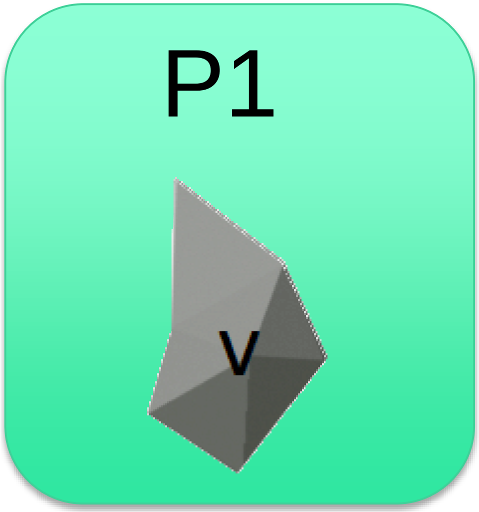
Merging Example with Two Producers

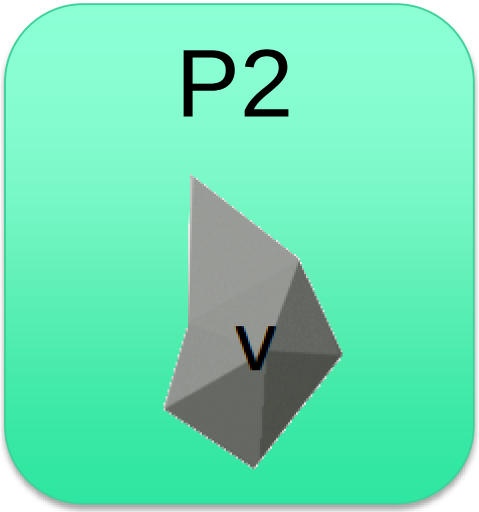


Merging Example with Two Producers

Merging Example with Two Producers

Merging Example with Two Producers





$$V = (10,12,10)$$

$$P1(V) = (10,15,10)$$
$$P2(V) = (12,17,10)$$

Merging Example with Two Producers

$$V = (10,12,10)$$

$$P1(V) = (10,15,10)$$
$$P2(V) = (12,17,10)$$

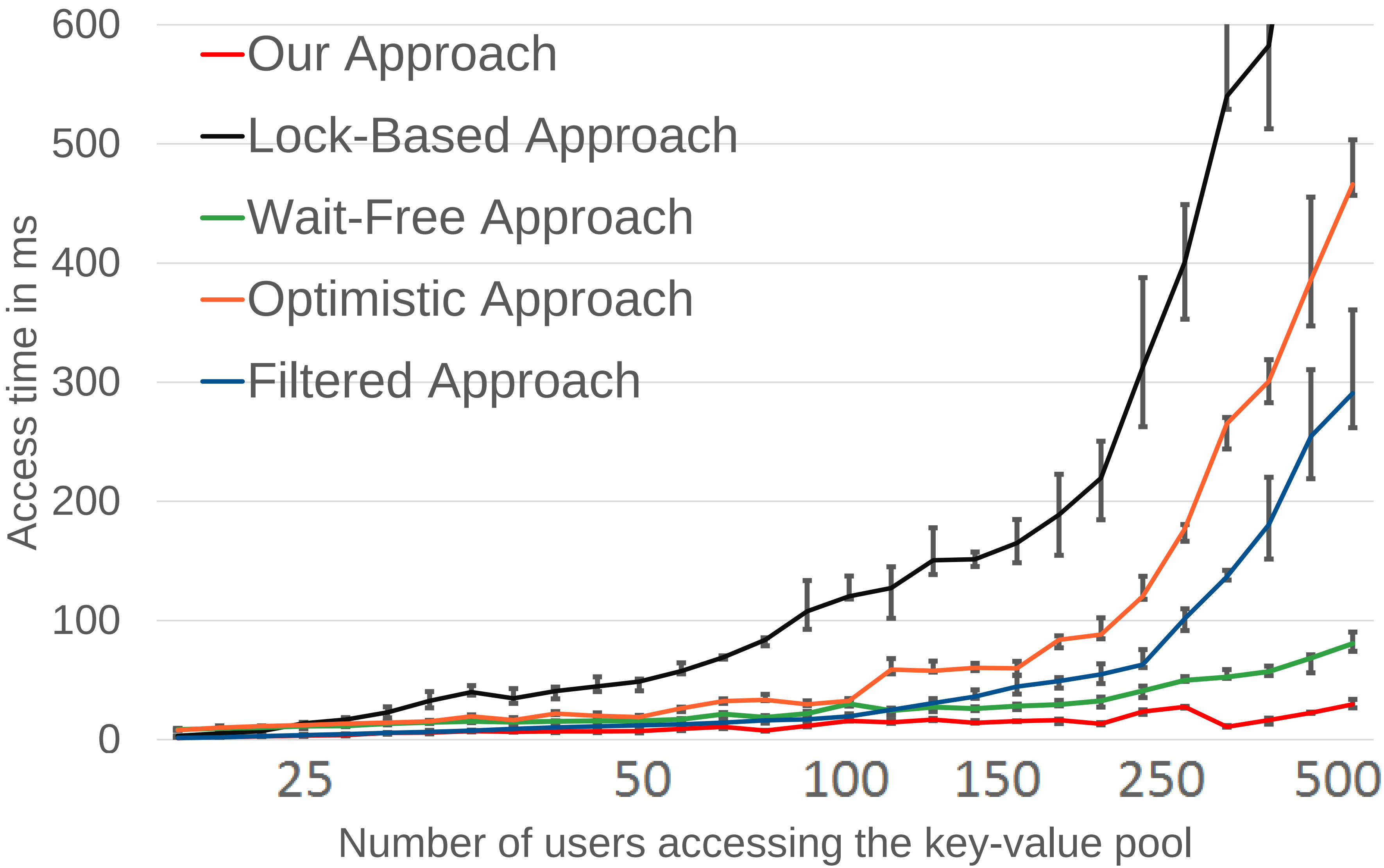
$$Merge(V) = (11,16,10)$$

Results

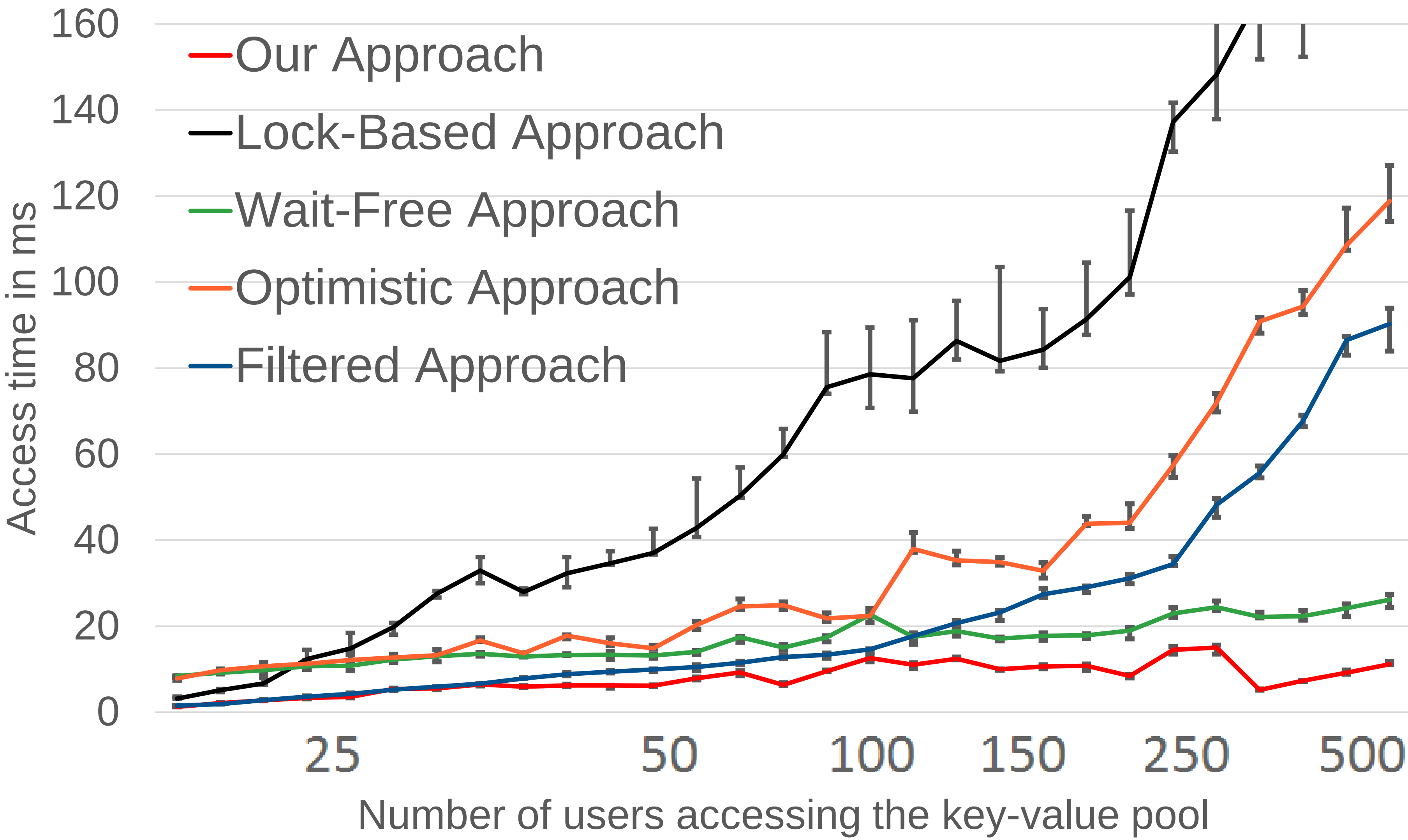
- Performance comparison with four competitors

1. Hash map with standard locking mechanisms from the boost library
   - Read and write operations are locking
2. Wait-free hash map based on previous work [Lange‘14]
   - Wait-free read and single wait-free write operations
3. Optimistic hash map based on [Wongwirat‘07]
   - No locking for read operations, rollback of transaction if transaction fail occurs
4. Filtered hash map based on [Li‘03]
   - Restriction on lock cast

Read (25%) & Write (75%) Operations



Read (50%) & Write (50%) Operations

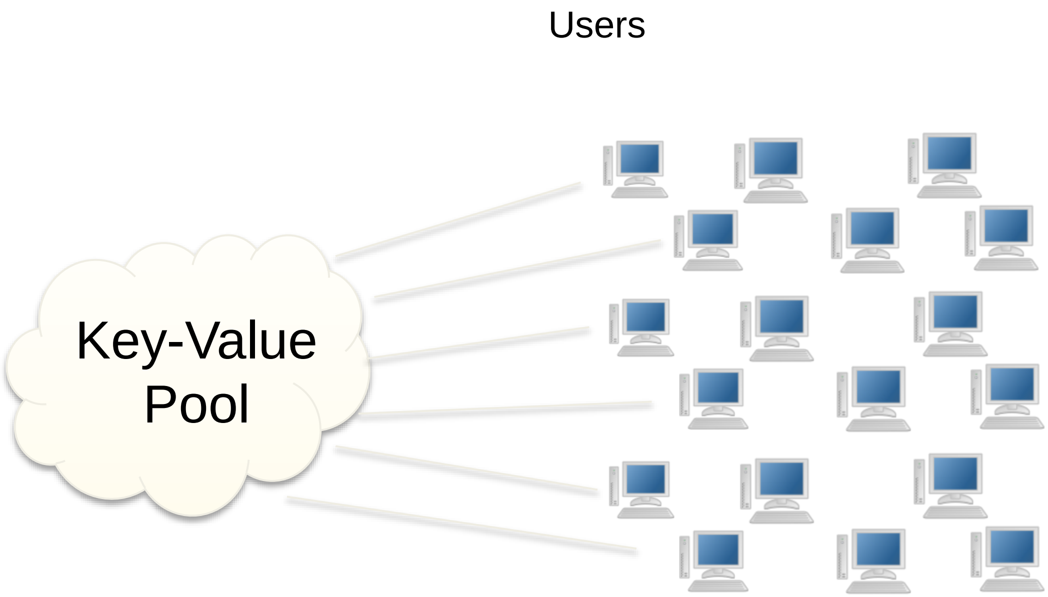


Conclusions

1. Scalable CCM for massively collaborative virtual environments
   - No deadlock, no starvation of user actions
   - Supports arbitrary non-blocking user interactions
2. Our novel CCM outperforms traditional approaches
   - Faster than a factor of 8-35
   - Less than 74% memory usage than our previous approach
3. Our novel CCM allows easy customization for many CVE applications
   - Data merge function can be defined for arbitrary purpose
   - Merge can also represent traditional approaches

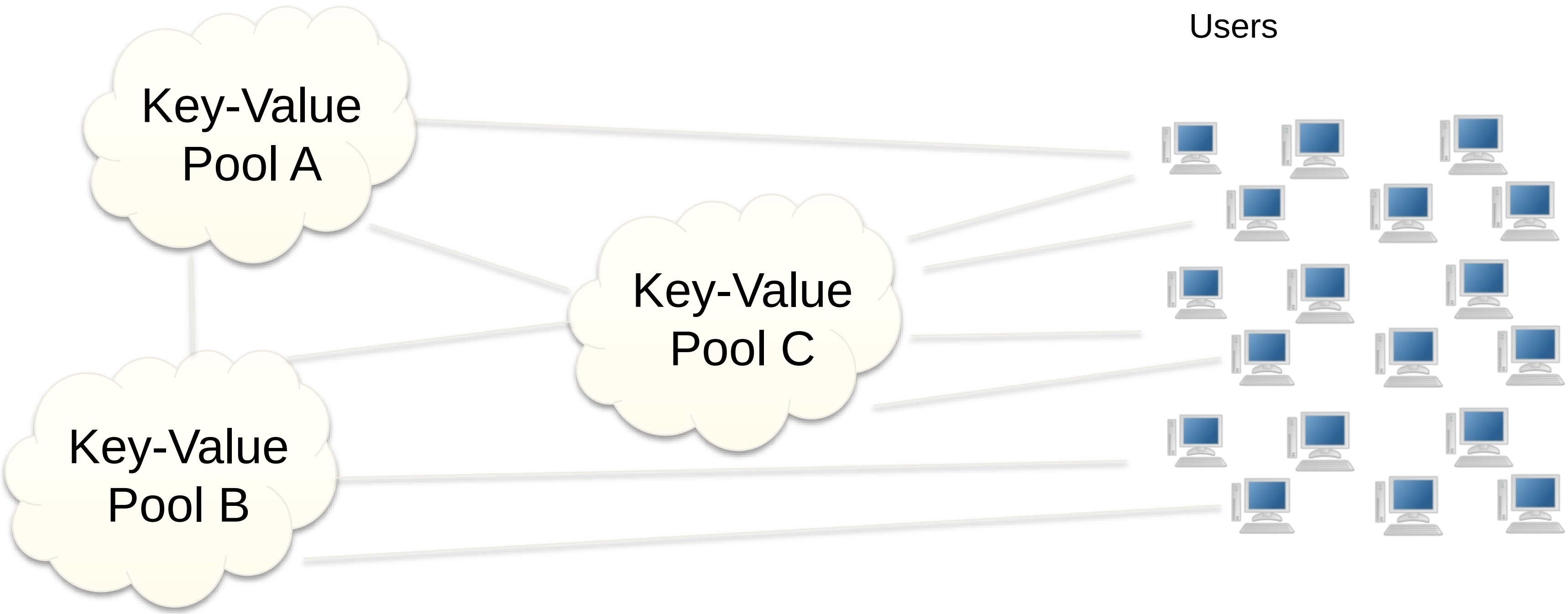
Future Work

- Distributed implementation and testing
  1. Key-value pool as central host



Future Work

- Distributed implementation and testing
  1. Key-value pool as central host
  2. Distributing key-value pools



Bremen

CG VR

Thank you for your attention

Questions?

Patrick Lange, Rene Weller, Gabriel Zachmann
{lange,weller,zach}@cs.uni-bremen.de

DLR

Funding by DLR, contract *50NA1318*

Cognitive Autonomous Navigation for Deep-Space Resource Mining
KaNaRiA
Universität Bremen · Universität der Bundeswehr München · DLR