

Eventual consistency is eventually not enough

Mehul A. Shah
work done while at HP

Acknowledgements

HP-KVS team: Eric Anderson, Jay Wylie, Xiaozhou Li, Joseph Tucek, Nitin Jain, Tom Hancock, Cian O'Driscoll, Bob Souza, and more

Sinfonia/Armonia team: Marcos Aguilera, Wojciech Golab, Alistair Veitch, Arif Merchant, Ben Sowell, Indrajit Roy, Stavros Harizopoulos, Nathan Binkert, and more

Introduction

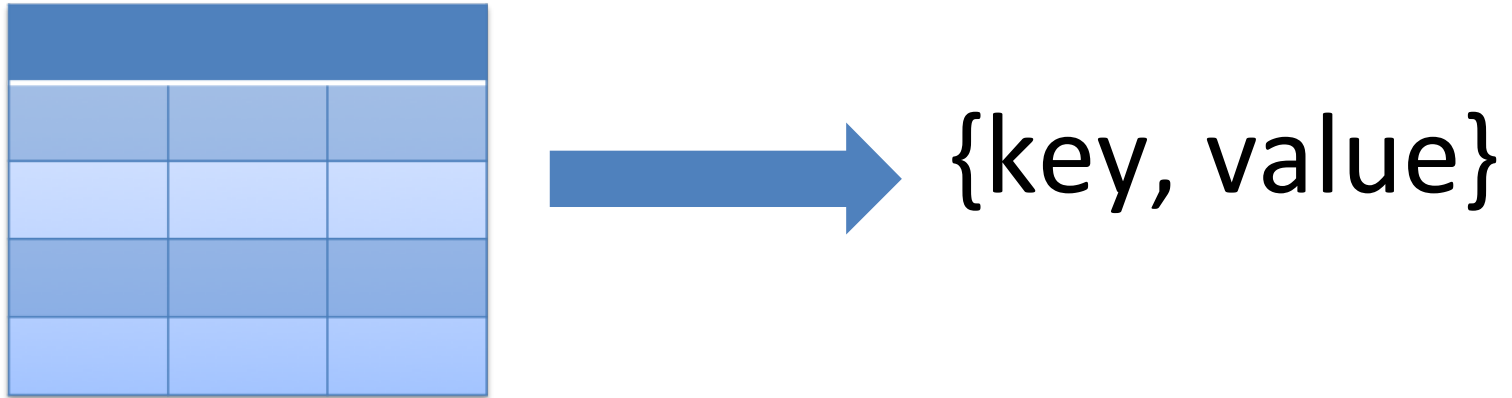
Apps need globally distributed,
scalable, 24x7 storage



Databases stink



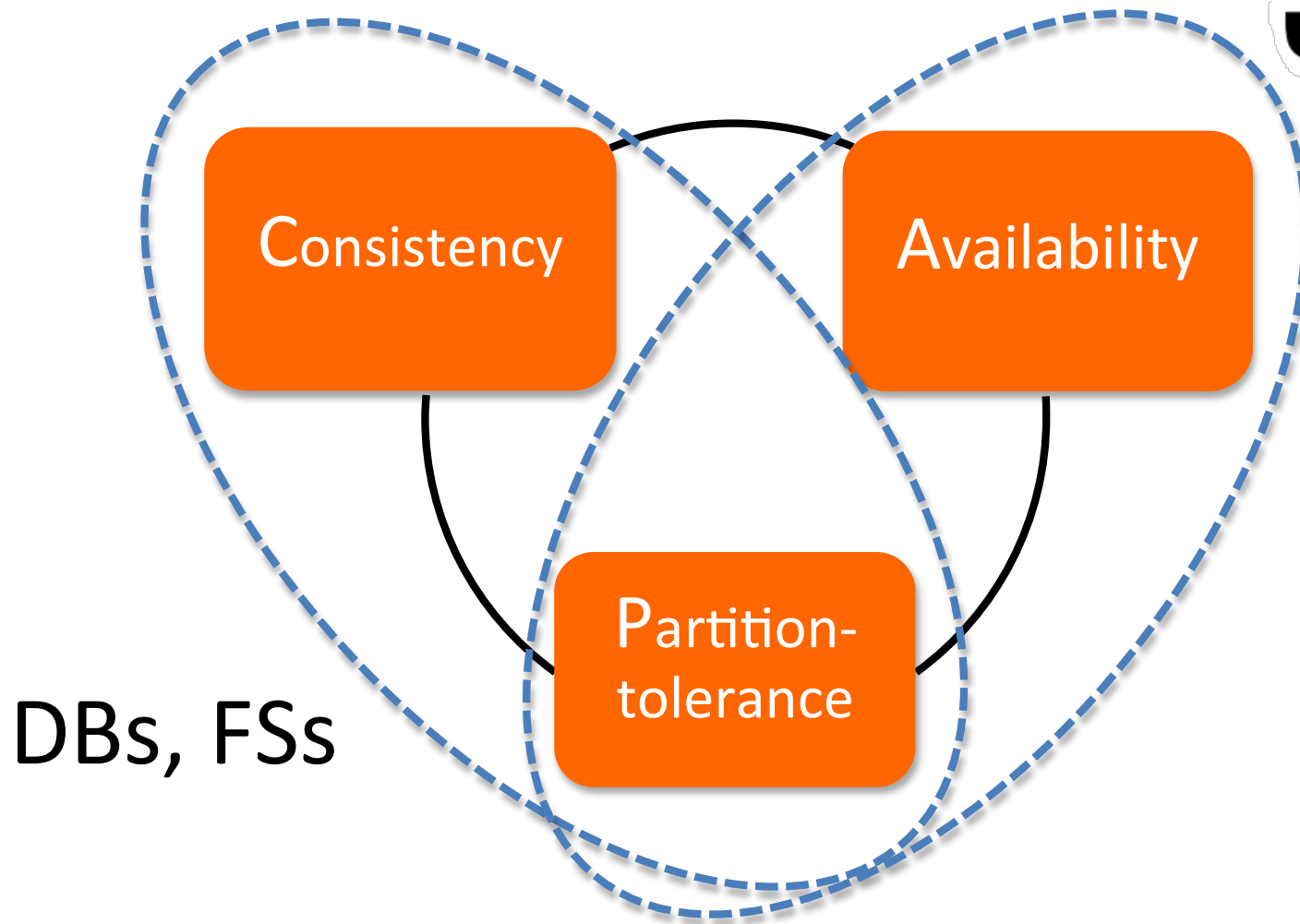
NoSQL reaction



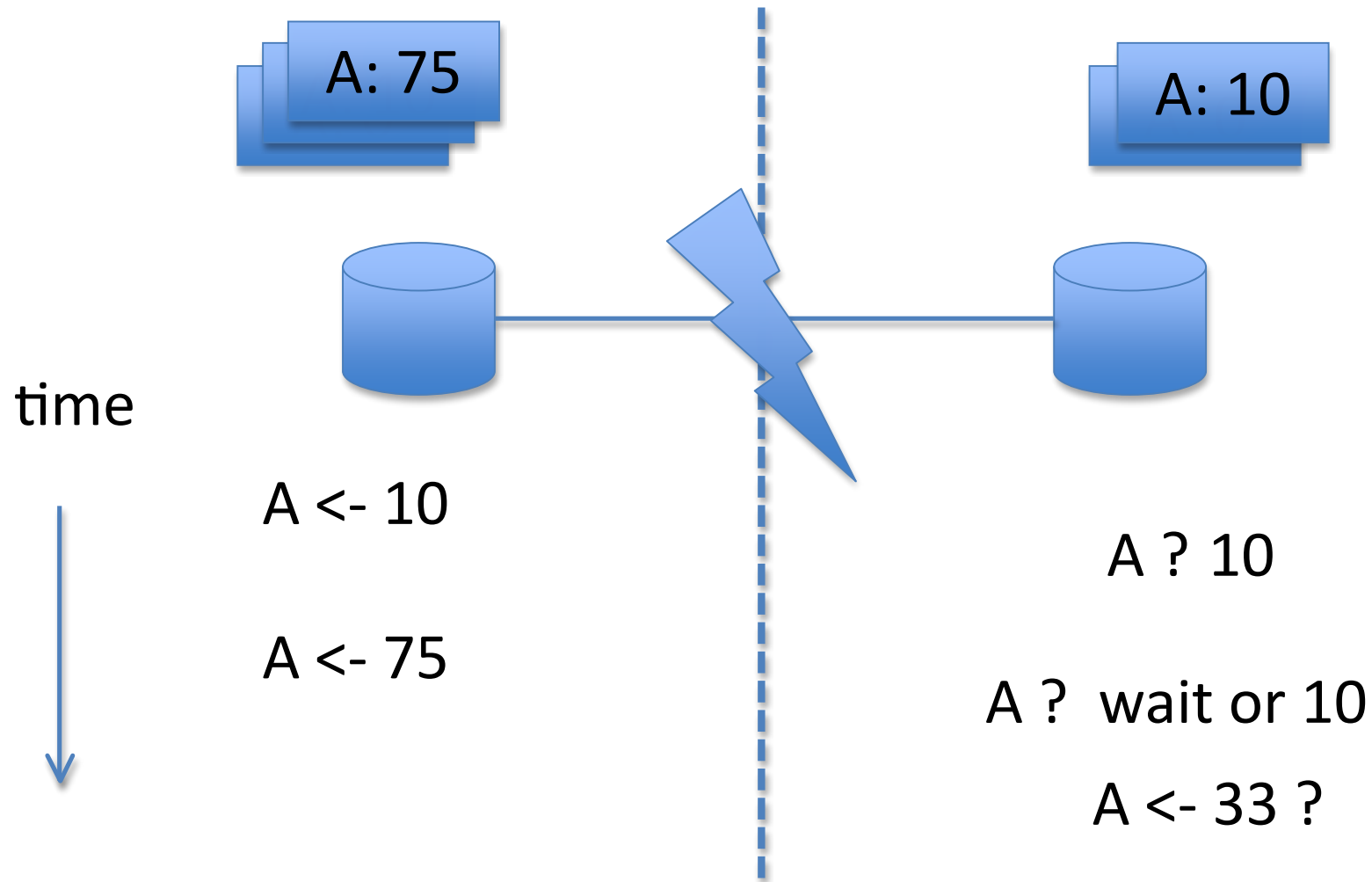
{key, value}



Brewer's CAP theorem



CAP explained



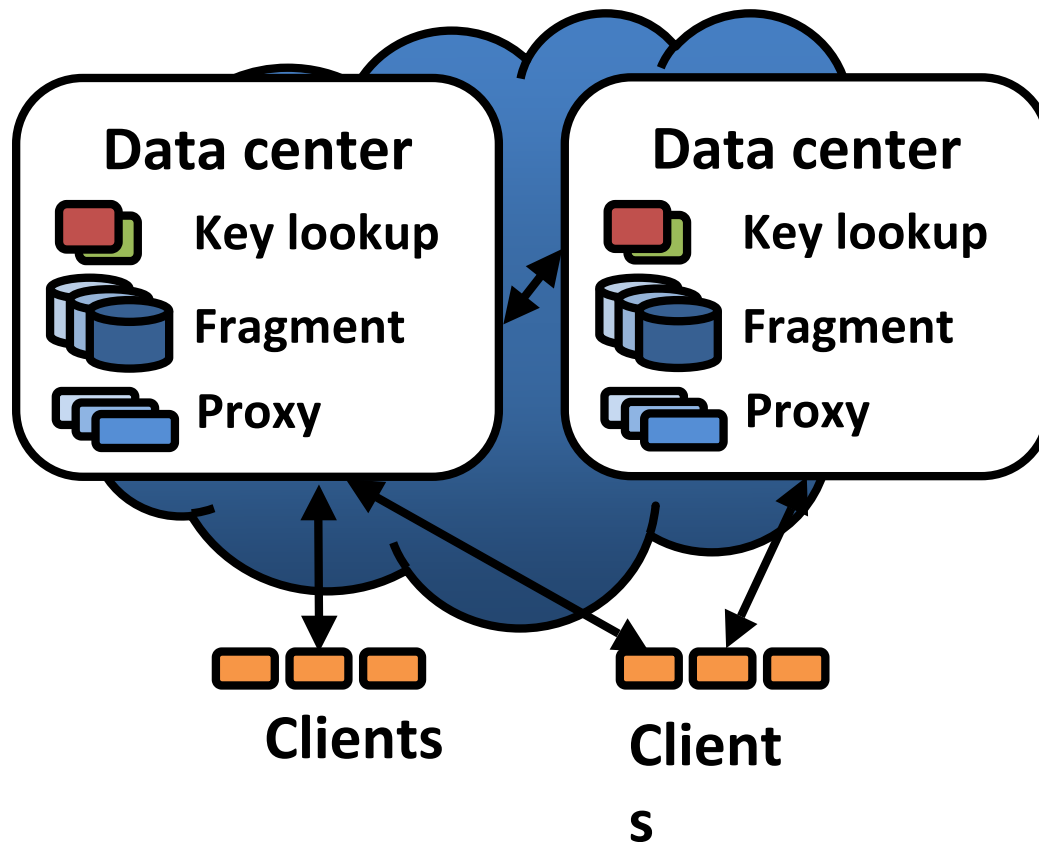
Myths

#1: Eventual consistency is enough

- HP-KVS experience
- Revisit CAP
- My ideal system

#2: Easy to add strong consistency later

HP-KVS overview

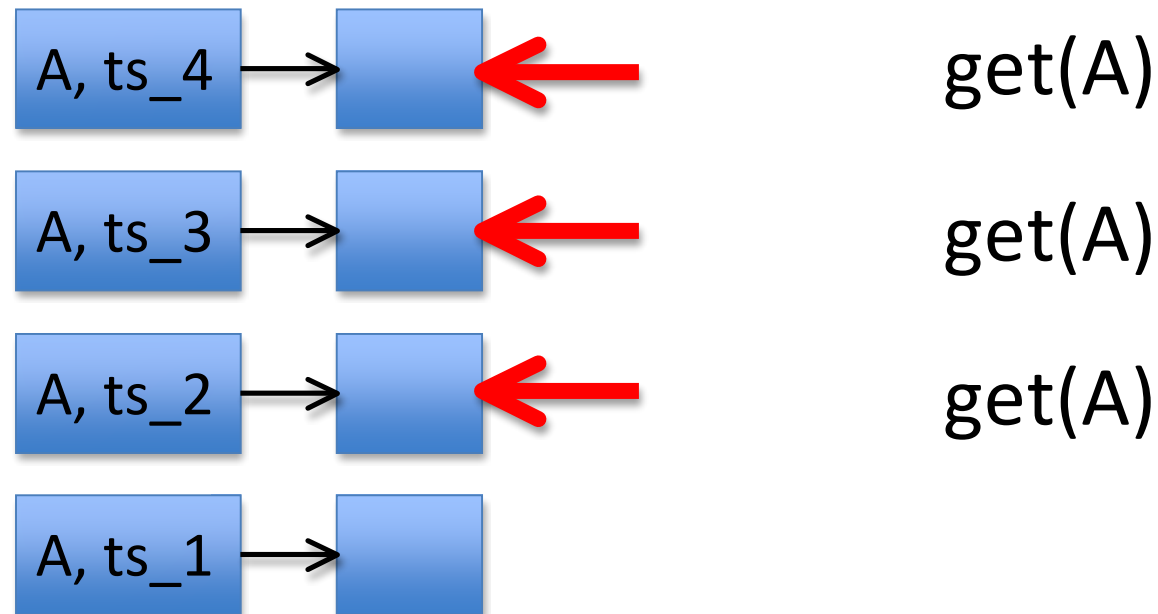


geo-distributed,
highly available,
low-cost

`put(key, value)`
`get(key) → value`
`enumerate(prefix
)`
`delete(key)`

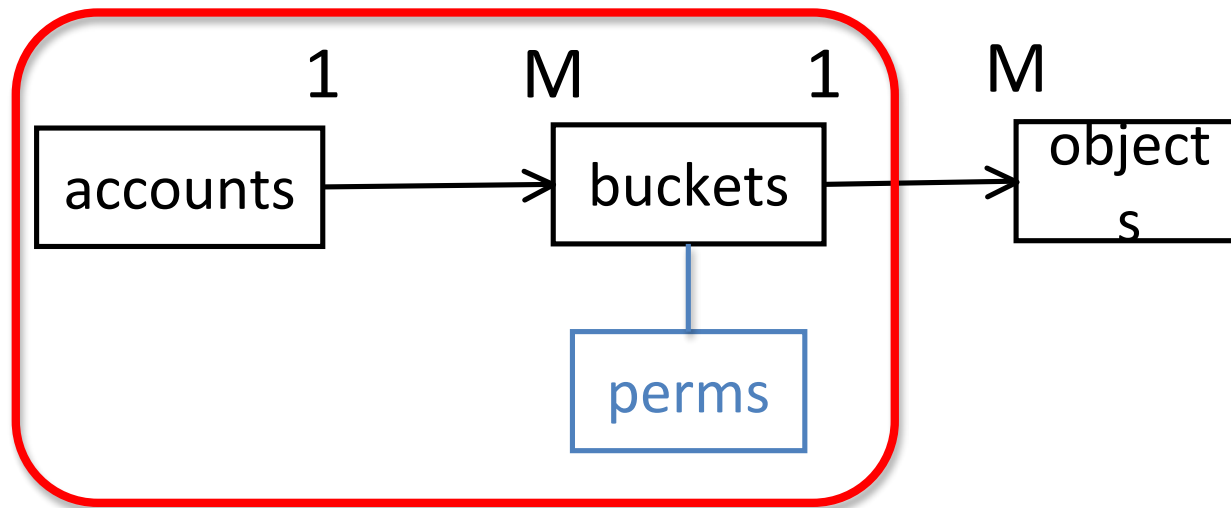
Eventual consistency

Versioned keys



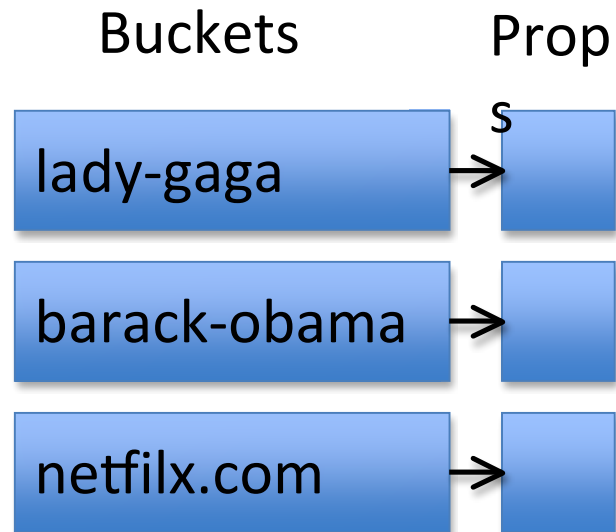
Requirements: many users

- S3-like interface on top
- How to handle meta-data ?

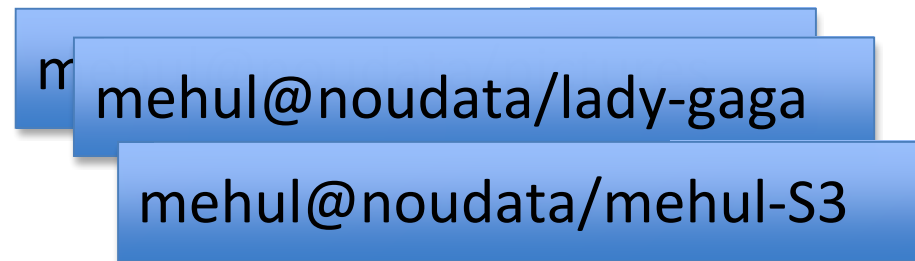


create(), delete()

Meta-data representation



Bucket listing per account



Objects in buckets



Myth #1: is enough

me

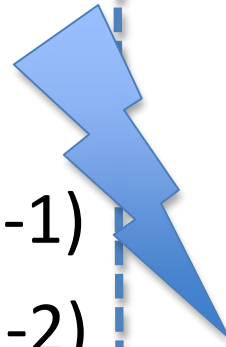
lady gaga

create("lady-gaga")

create("lady-gaga")

putObj("lady-gaga",diya-1)

putObj("lady-gaga",diya-2)



Atomic conditional update (ACU)

- Need atomic: get/test + put
 - Cannot do this without a test!
- Multi-key: consistent per-user bucket lists

ACU: {*verify set*, *update set*}
verify set: {key, exists?, latest version}
update set: {key, value}

Do we need more ?

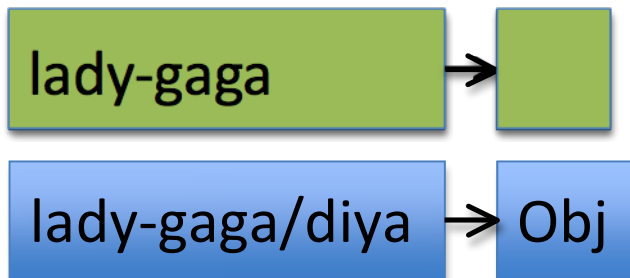
- put/get object highly available -> weak
- put/get object checks bucket permissions
- Mixing strong (ACU) and weak ops (put/get)
 - independent objects -> no concern
 - same object, multi-object ?

Subtle interactions

Weak: check bucket perm
add object diya

putObj("lady-gaga", diya)
delete("lady-gaga")

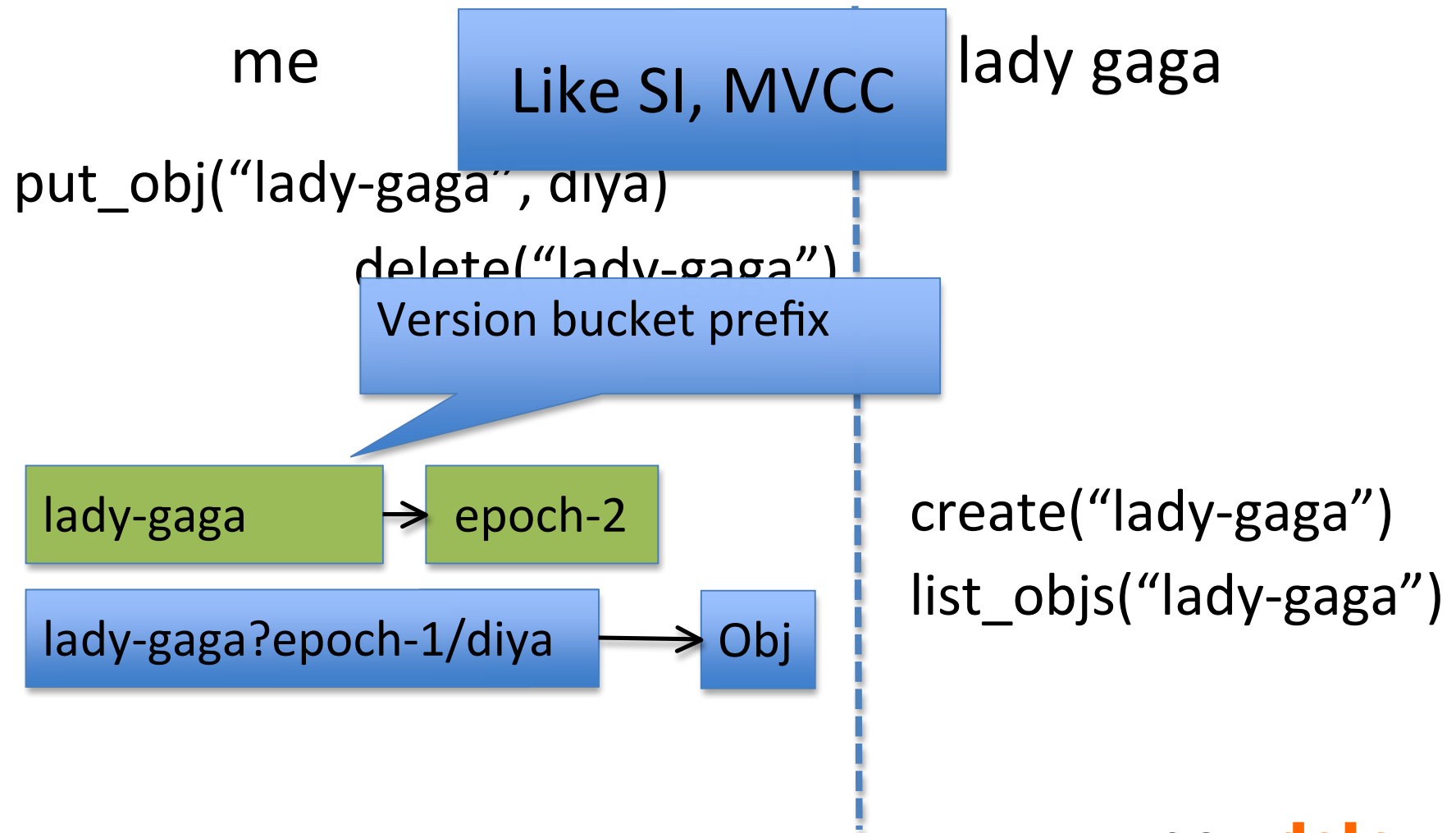
ACU: checks no objects,
removes bucket key



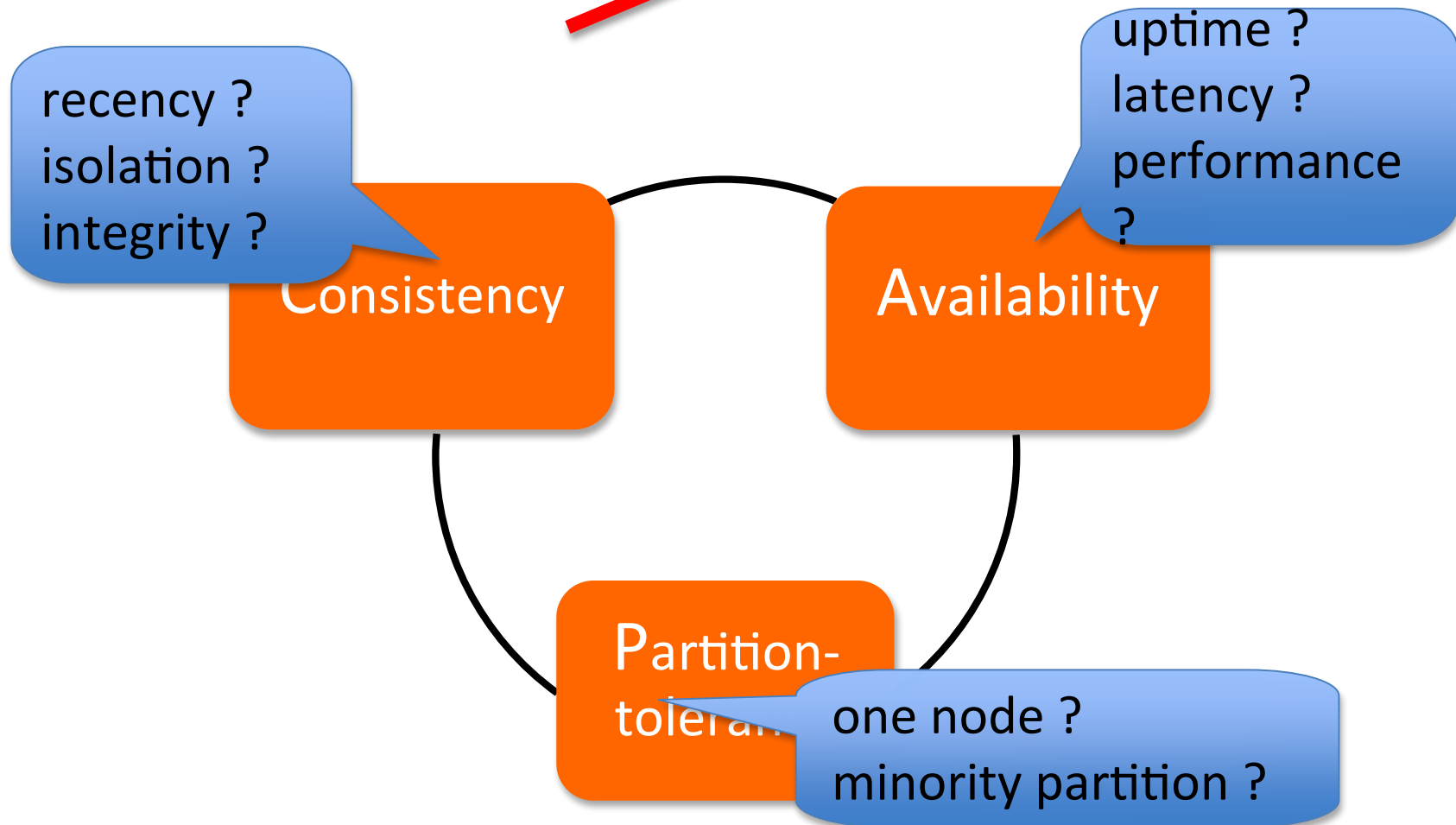
create("lady-gaga")
listObjs("lady-gaga")

Returns phantom object

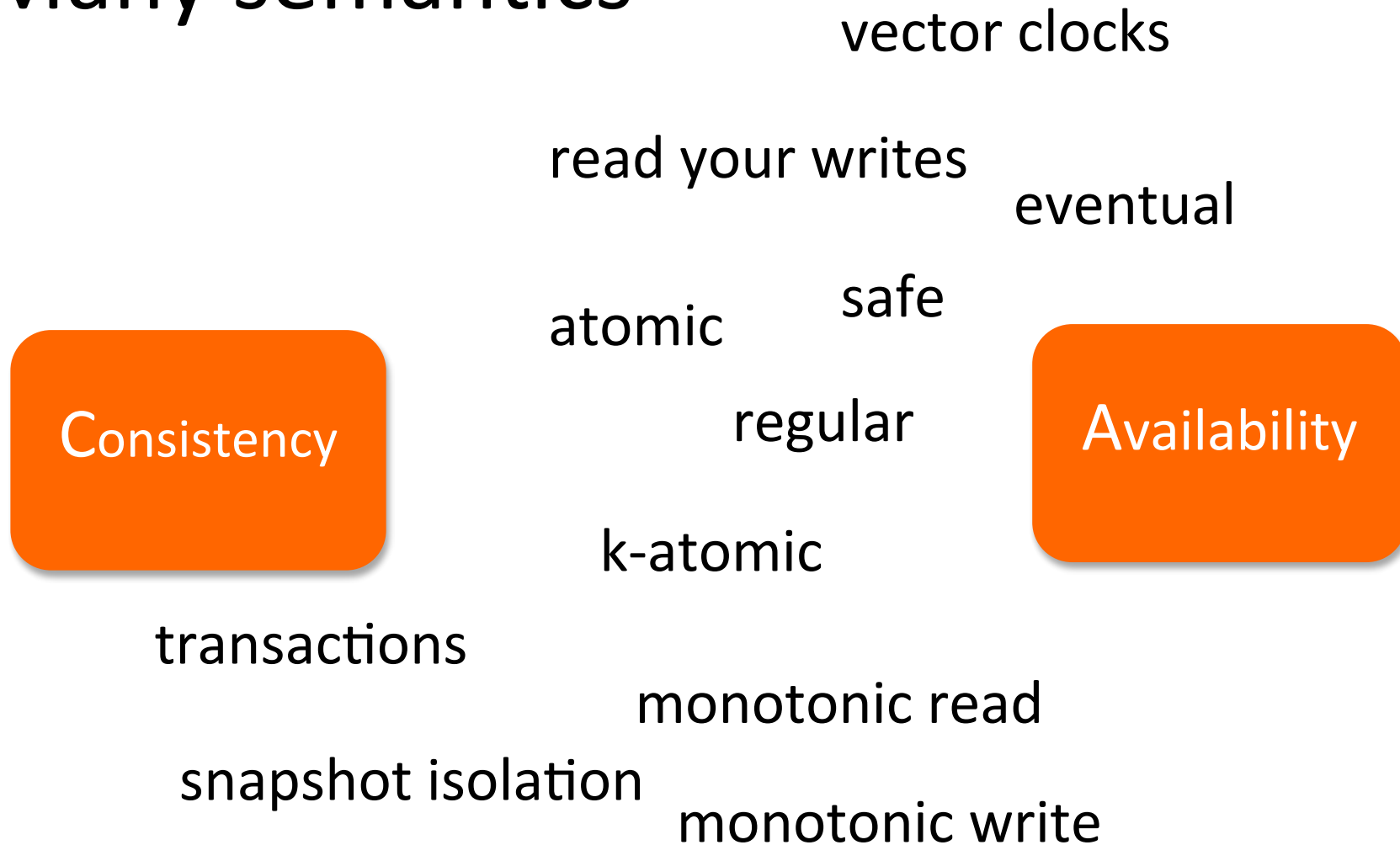
Subtle interactions



Brewer's CAP theorem ~~Principle~~



Many semantics



Spectrum in single system



Like isolation levels

- Easy to understand from interface/use-cases
- Consistency levels are compatible
- Spans fault-space and differentiated service
 - rack, WAN partitions, single key, multi-key, etc.
 - Consistability [Anderson et al.]

Myth #2: easy to add SC later

- Eventually consistent is just as hard or harder
 - specification not trivial
 - ensure invariants are eventually true
 - HP-KVS: convergence, purging, abandonment, ...
- How do we add ACU ?

Option #1

Not enough underneath, limiting



Option #2

Pro: industrial strength software

Con: two different interfaces and systems

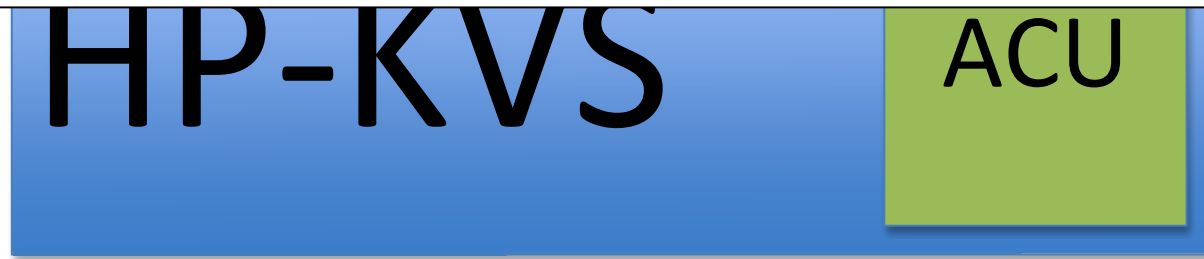
Con: availability and durability limited by DB



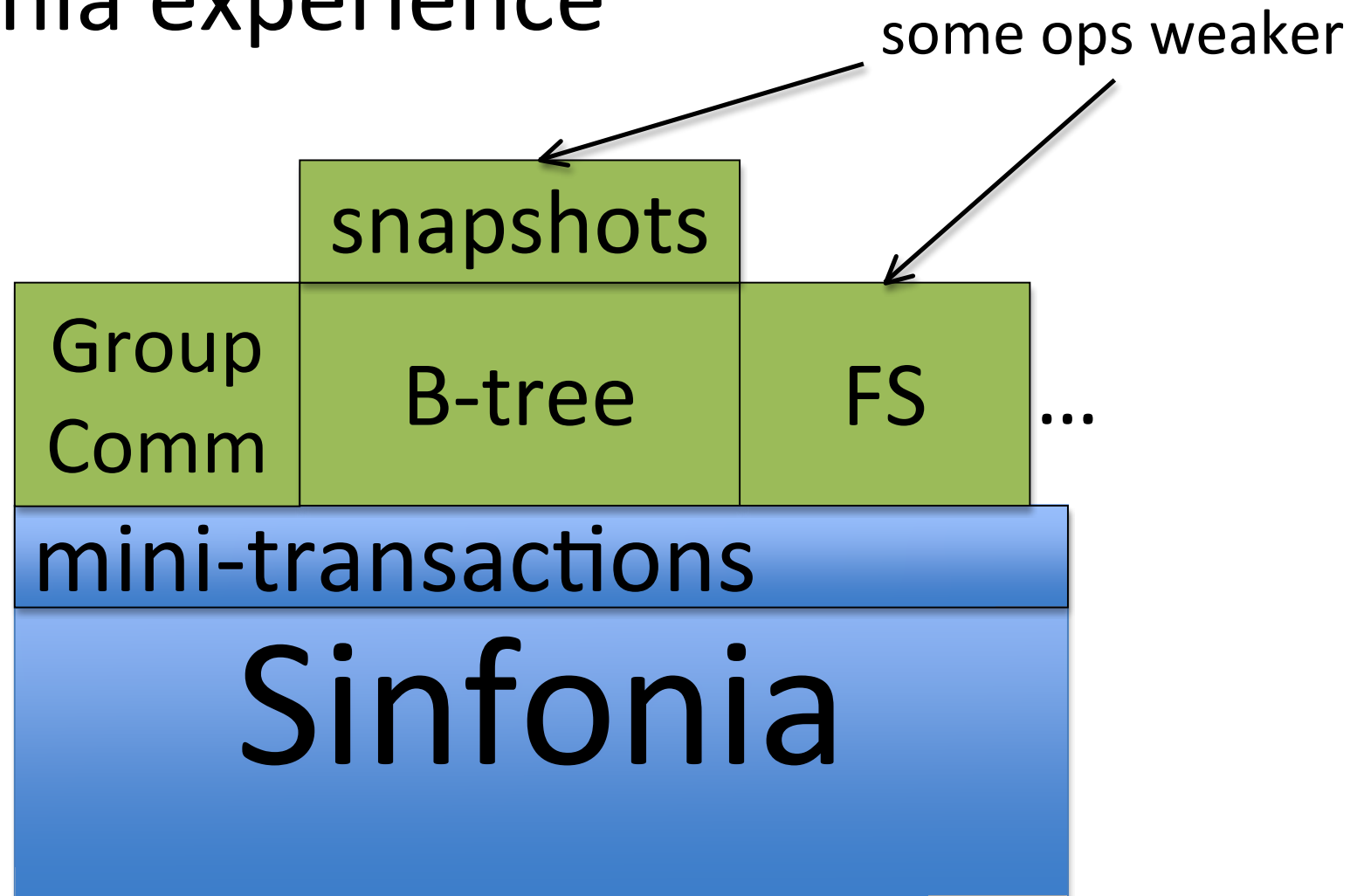
Option #3

Best option

Must reason carefully about interactions with eventual consistency protocols



Sinfonia experience



Starting from scratch ?

- Build a strongly consistent (distributed) core
- Relax consistency when and where needed
- Easier to relax later than strengthen
 - E.g. combining strongly consistent cores
 - E.g. removing protocol steps: atomic vs. regular

Related Work

- ANSI isolation levels
- Acta framework [Chrysanthis + Ramamritham]
- PACELC [Abadi]
- CONITs [H. Yu et al.]
- Consistability [Anderson et al.]
- PNUTs [Cooper et al.]
- and many more ...

Conclusion

- Eventual consistency needed
 - hard to understand
 - not enough
- Want integrated store with more options
- New world with new opportunities
 - time to revisit with rigor