codecentric
agile software factory

DARF'S EIN WENIG GRÖSSER SEIN?

Architekturmuster für hochskalierbare Systeme

Uwe Friedrichsen – Berlin Expert Days 2012 – Berlin – 29. März 2012

ÜBER MICH …

Name: Uwe Friedrichsen

Berufserfahrung: Relativ vielfältig und lang

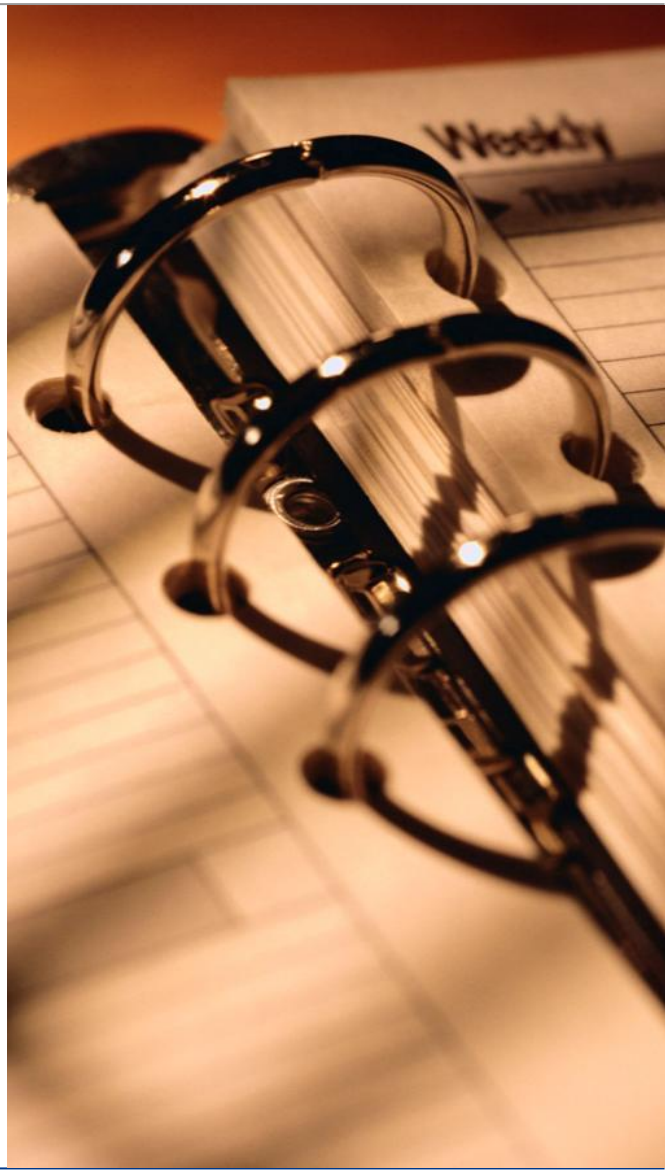
Schwerpunkte:

- Teams, Projekte und Systeme zum Erfolg führen – mit einem speziellen Fokus auf Architektur und Agilität
- Ganzheitliches Denken, Ideen und Konzepte verknüpfen, Leute zum Nachdenken bringen
- Neue Konzepte & Technologien

Position: CTO bei codecentric AG

AGENDA

- Basics of scalability
- Dimensions of scalability
- Design for high scalability
- Choose the right tool
- More stuff …
- Summary



Strive for Simplicity

The system should be made as simple as possible (- but no simpler)


Scale out, not up

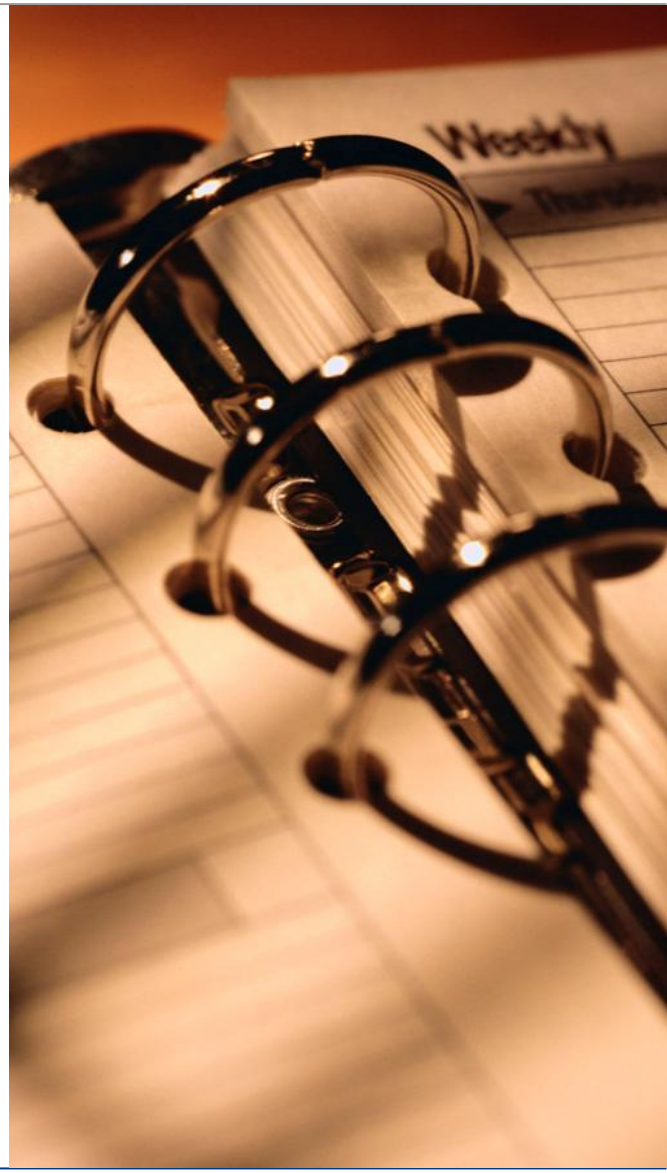

Be stateless


Share nothing

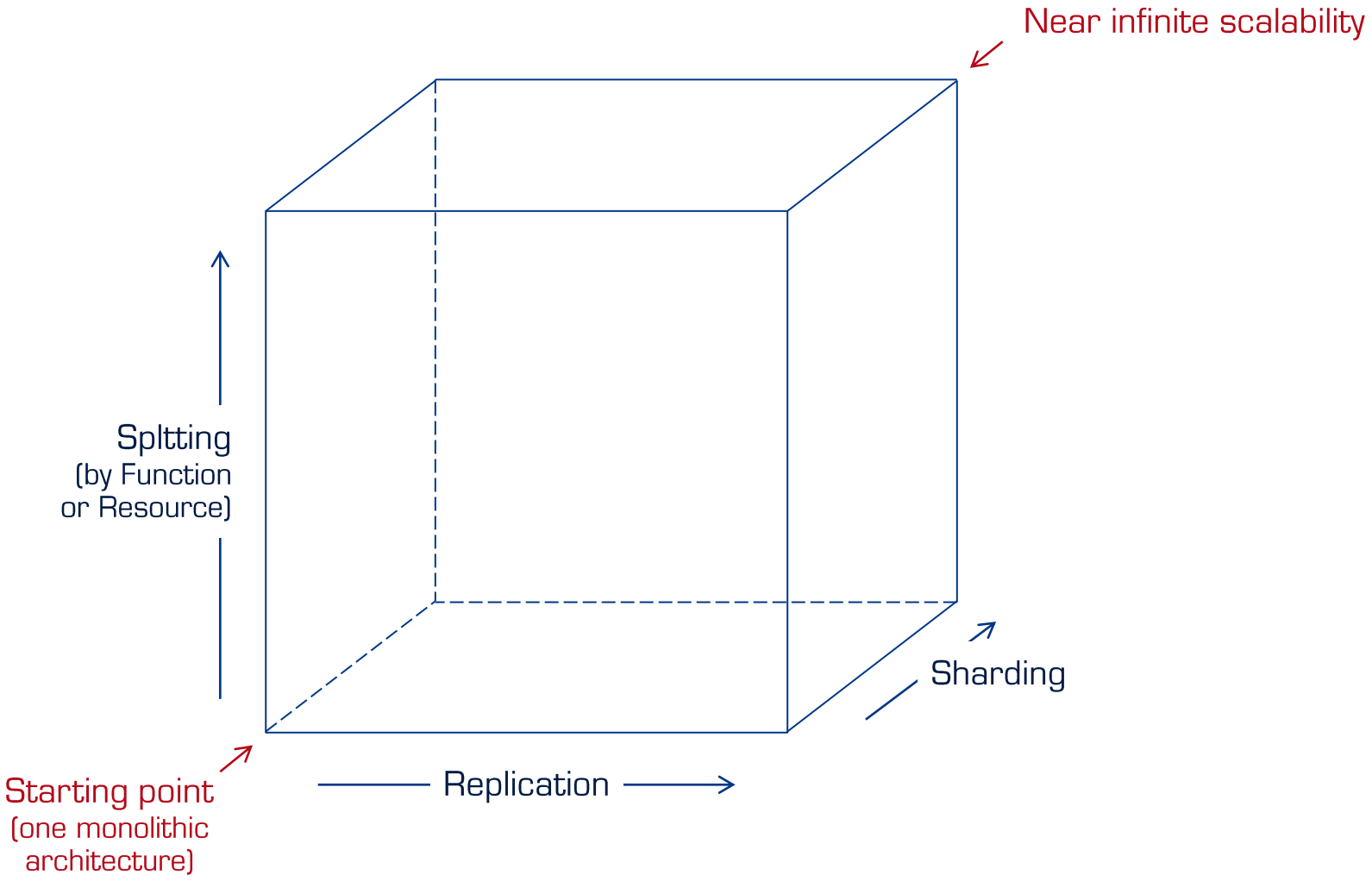

Communicate
asynchronously

AGENDA

- Basics of scalability
- Dimensions of scalability
- Design for high scalability
- Choose the right tool
- More stuff …
- Summary



DIMENSIONS OF SCALABILITY



Source: [1]

REPLICATION

When to use

- Scaling of transactions, not data
- High read to write ratio

How to implement

- Clone services and distribute calls via load balancer
- Use replication features of infrastructure components (database, app-server)

Related Concepts

- Load balancer & shared nothing units
- Load balancer & stateless nodes & scalable storage
- Master slave replication
- Masterless replication

Tradeoffs

- Doesn‘t scale well with high write rates
- Doesn‘t work for unpredictable data volumes

SPLITTING

When to use

- Very large data sets of loosely coupled data
- Large complex systems with loosely coupled functionality

How to implement

- Split up by noun (data) or verb (function)
- Best implemented with shared nothing units and/or anynchronous communication

Related Concepts

- Service oriented architecture (SOA)
- Resource oriented architecture (ROA)
- Pipe & Filter

Tradeoffs

- Data and/or functions aren't always suitable for splitting
- No tool support (as for replication or sharding)
- Long functional chains make system less reliable

SHARDING

When to use

- Very large data sets of tightly coupled data
- Unpredictable data volume, rapidly growing data sets

How to implement

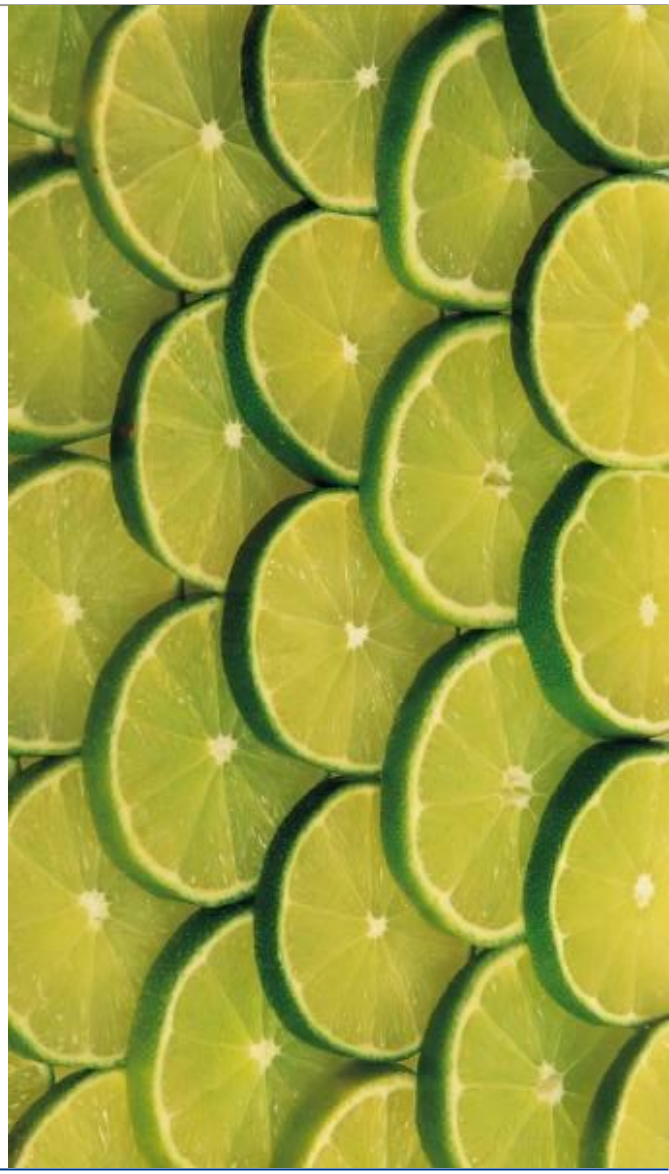
- Use sharding feature of your database component

Related Concepts

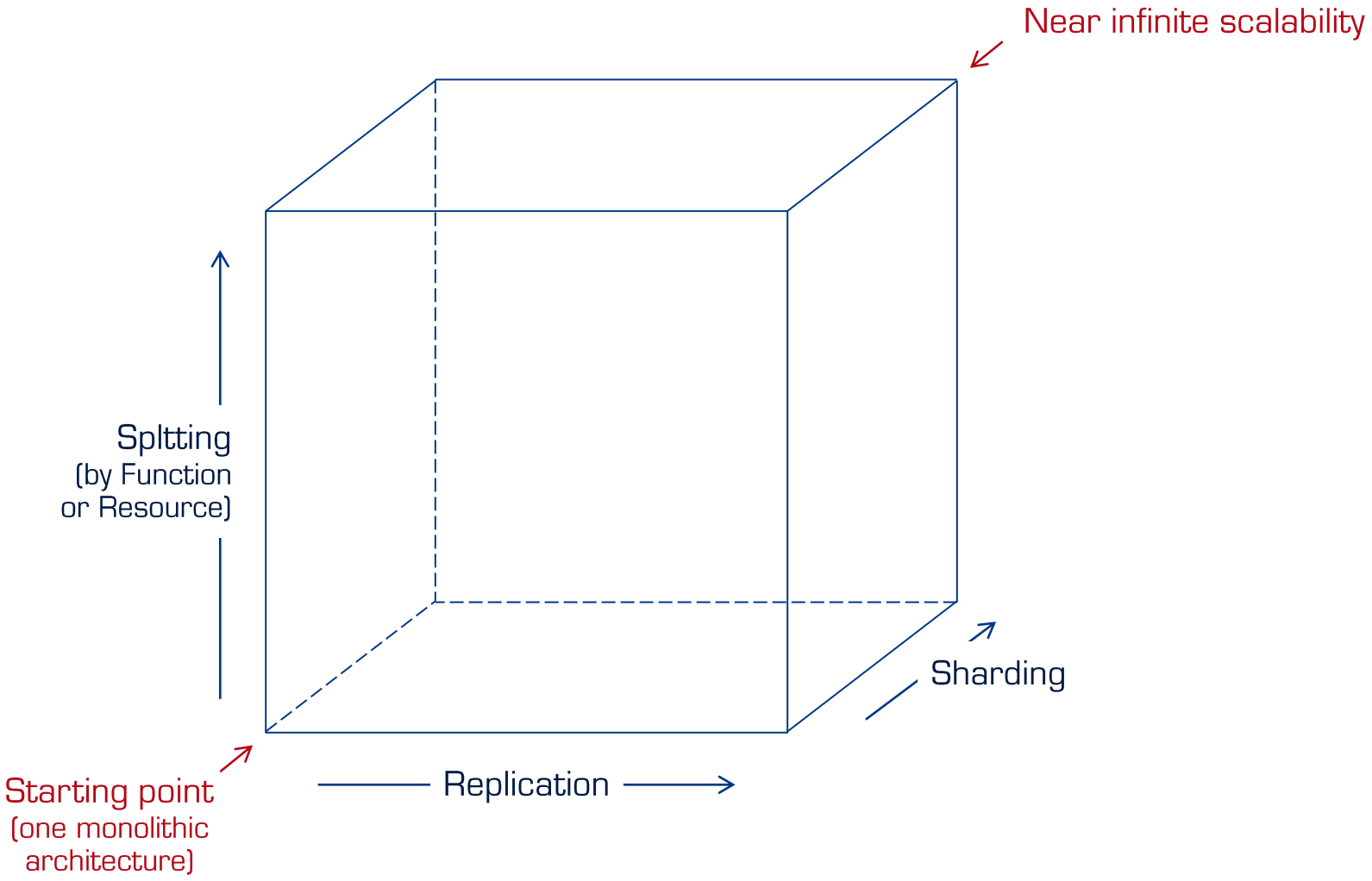
- Consistent Hashing
- Scatter & gather (MapReduce)

Tradeoffs

- Consistency is relaxed (especially for indices)
- Simple sharding does not work well for rapidly growing data sets
- Expensive in terms of required computing resources



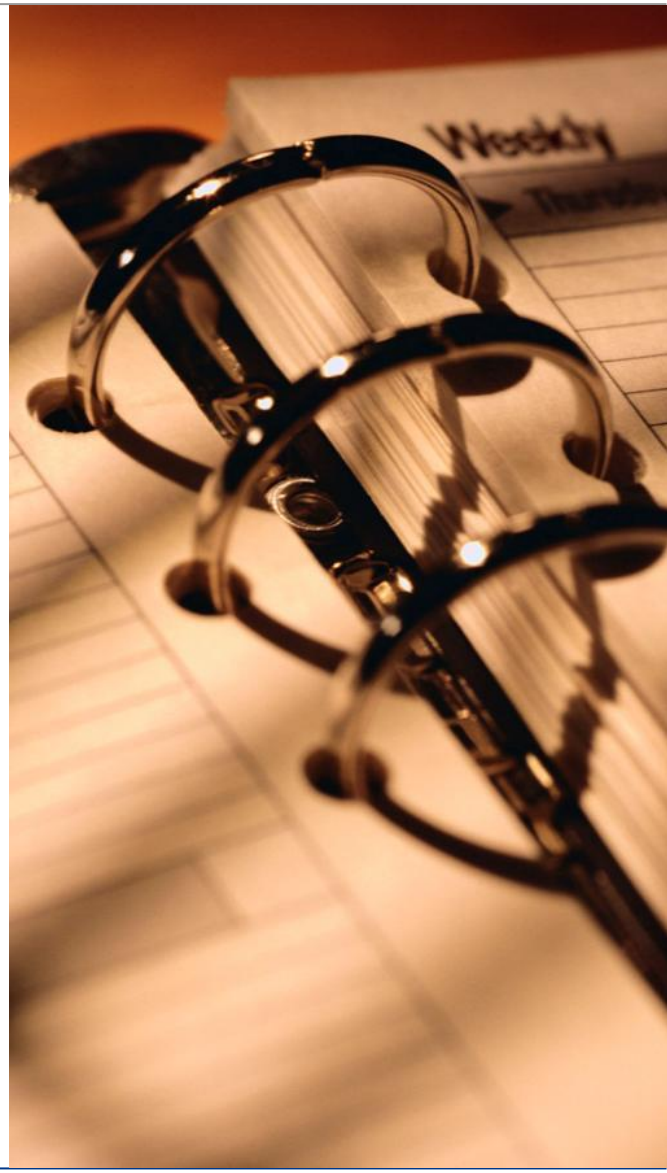
DIMENSIONS OF SCALABILITY



Source: [1]

AGENDA

- Basics of scalability
- Dimensions of scalability
- **Design for high scalability**
- Choose the right tool
- More stuff …
- Summary




Relax temporal constraints

RELAX TEMPORAL CONSTRAINTS

When to use

- Distributed data sets and the CAP theorem gets into the way
- Availability is more important than immediate consistency

How to implement

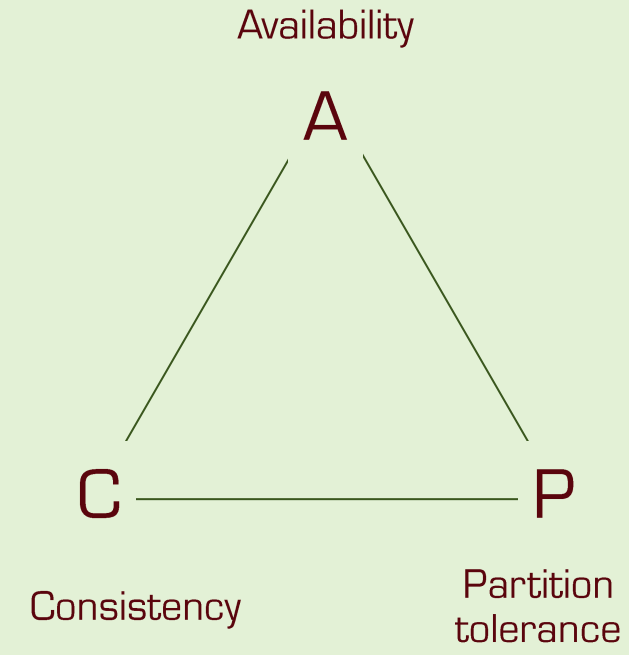
- Consider carefully your availability and consistency requirements
- Use a datastore that supports your requirements

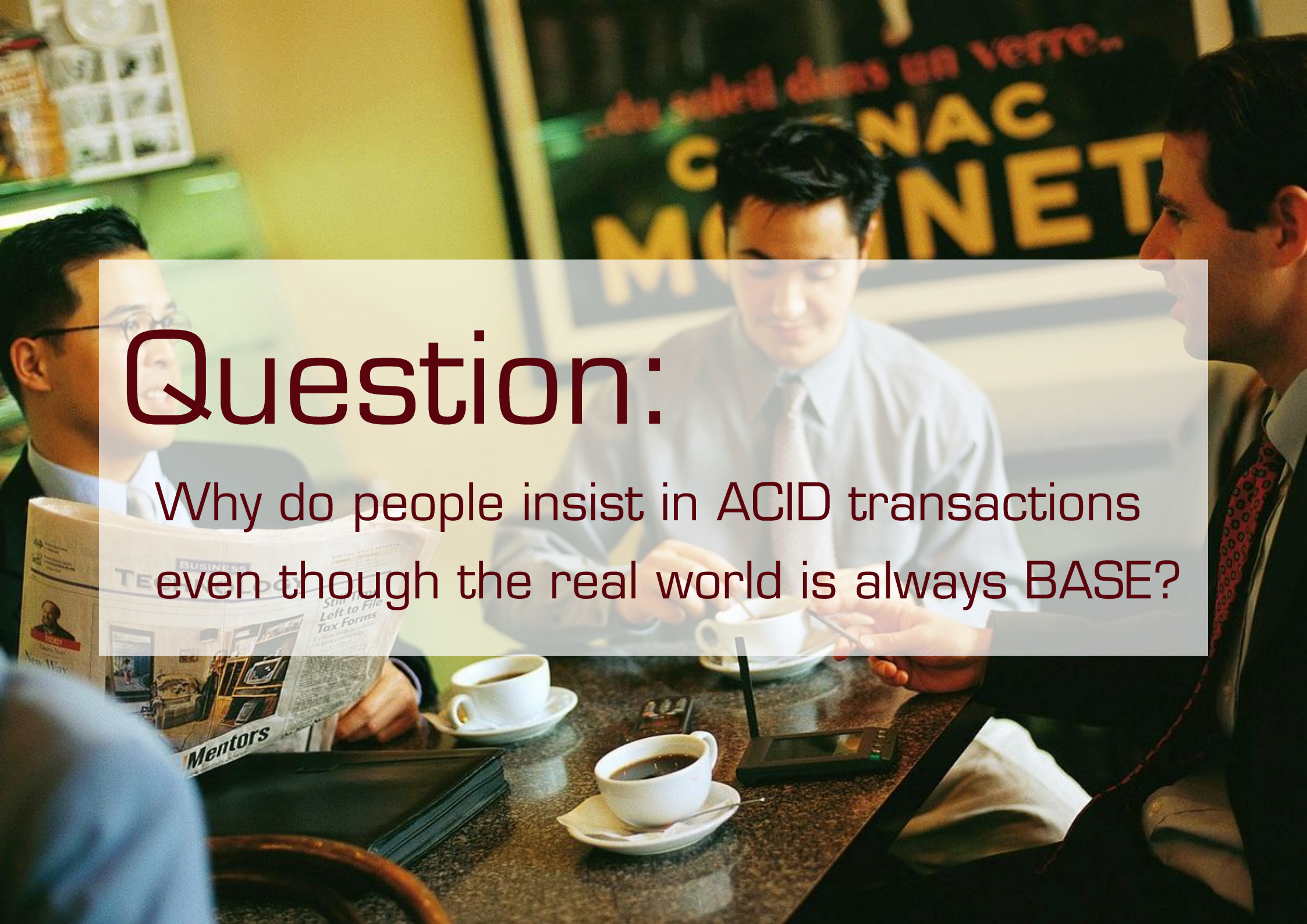
Related Concepts

- Quorum
- Harvest & Yield
- Hinted handoffs

Tradeoffs

- BASE consistency, no ACID consistency
- Application might need to handle temporal inconsistencies itself





Question:

Why do people insist in ACID transactions even though the real world is always BASE?

Cache as cache can

CACHE AS CACHE CAN

When to use

- Short response times and/or high throughput are important
- High read to write ratio

How to implement

- Use caches built-in to products currently in use
- Insert dedicated cache layers into your architecture
- Plan – Do – Check – Act

Related Concepts

- Content Delivery Networks
- HTTP Caching – Expires and ETag headers
- Distributed caches

Tradeoffs

- Caches may become stale
- Balance between response time and additional resources


Keep dynamic data closer to the compute
and static data closer to the end-user

DYNAMIC AND STATIC DATA

When to use

- Optimized response times with large heterogeneous data sets
- Network bandwidth and/or transfer costs are an issue

How to implement

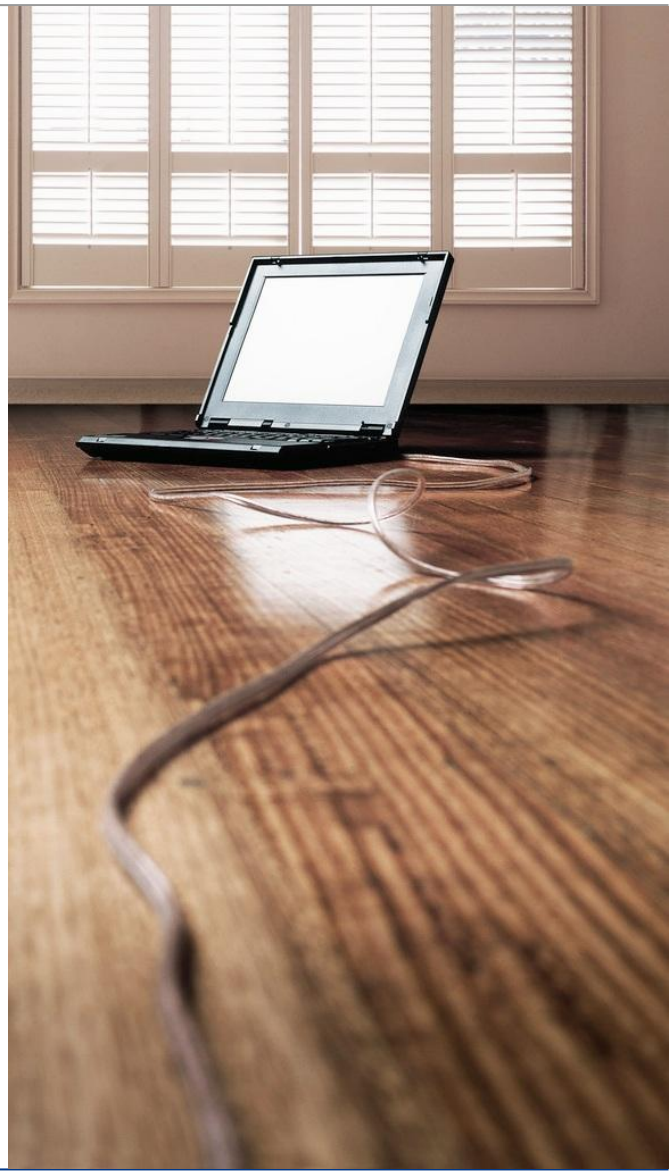
- Analyze your data and identify static and dynamic portions
- Store dynamic data on same nodes where processing logic is
- Move processing logic to the data storage
- Use frontend caches for static data

Related Concepts

- Shared nothing
- Scatter & gather
- Content Delivery Networks
- HTTP Caching – Expires and ETag headers

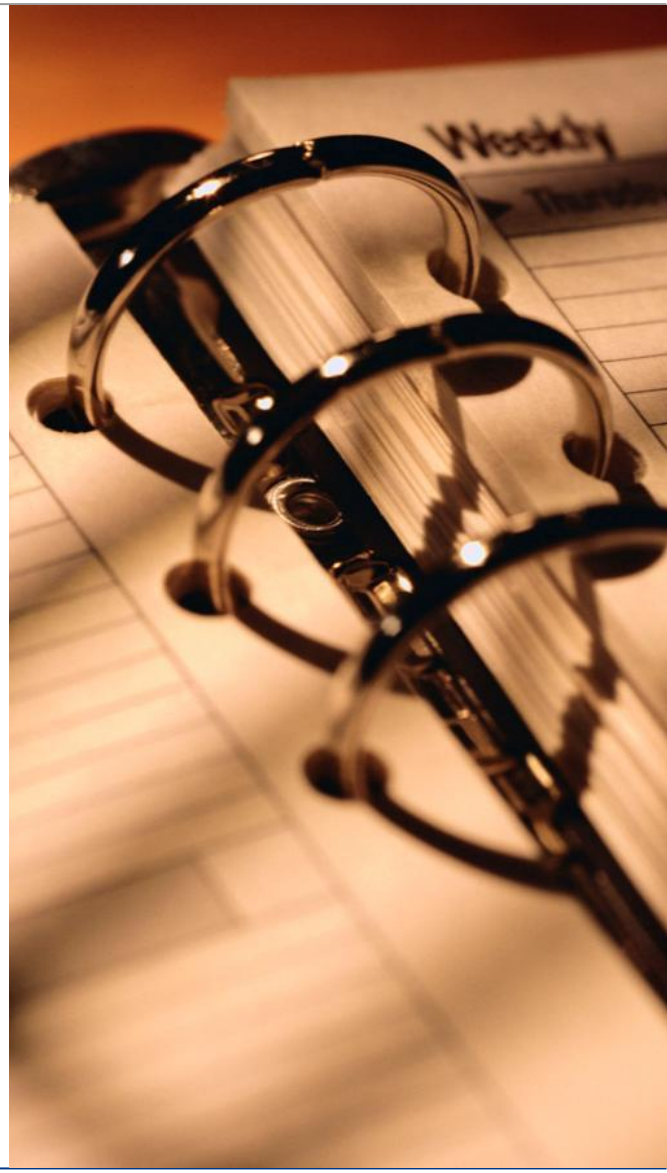
Tradeoffs

- Must be implemented explicitly

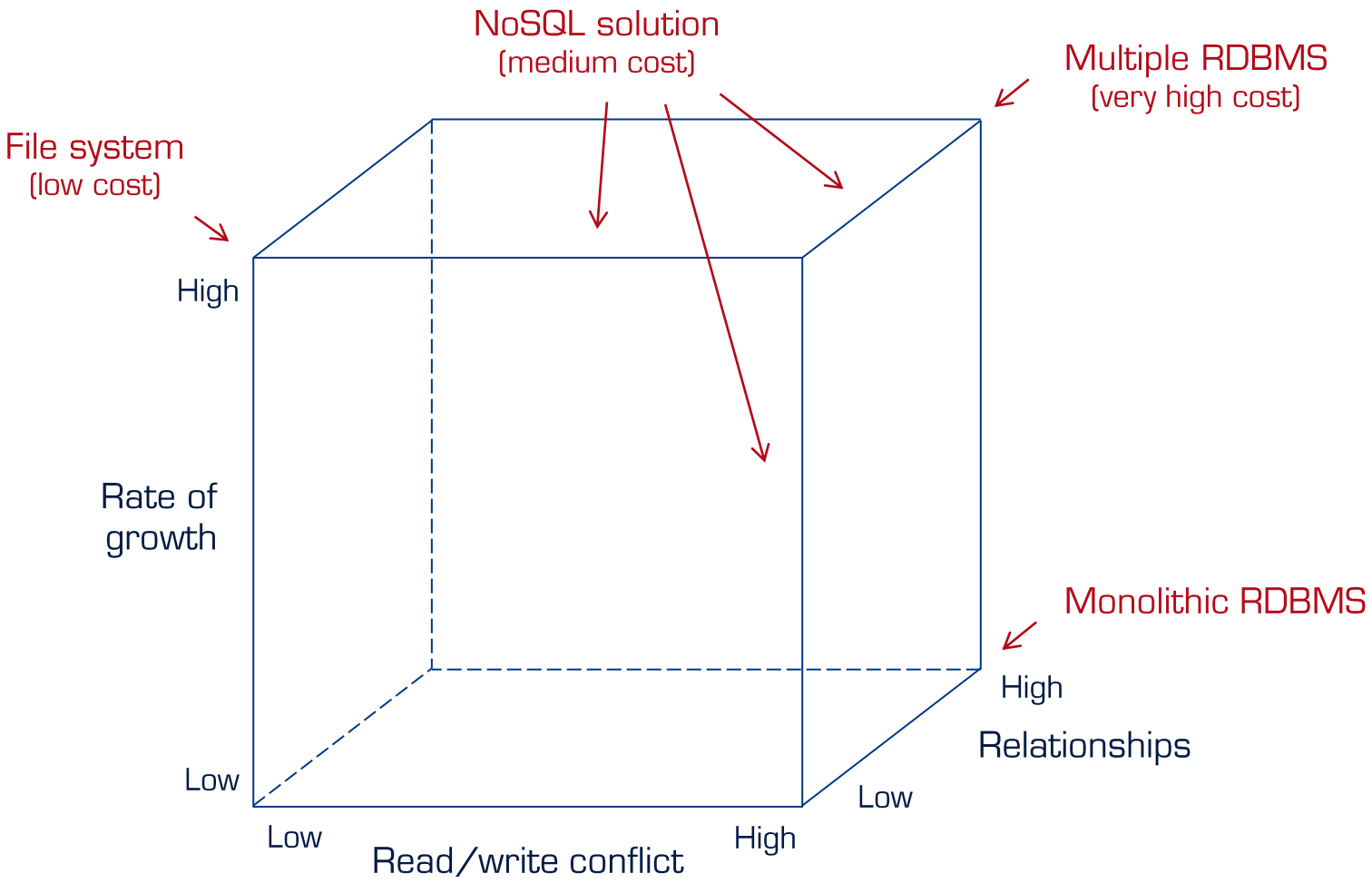


AGENDA

- Basics of scalability
- Dimensions of scalability
- Design for high scalability
- **Choose the right tool**
- More stuff …
- Summary



STORAGE TECHNOLOGY DECISION CUBE



Source: [1]

STORAGE TECHNOLOGIES

File system

- Easy to use, scales extremely well
- Can handle large entries well
- Poor support for relations and concurrency

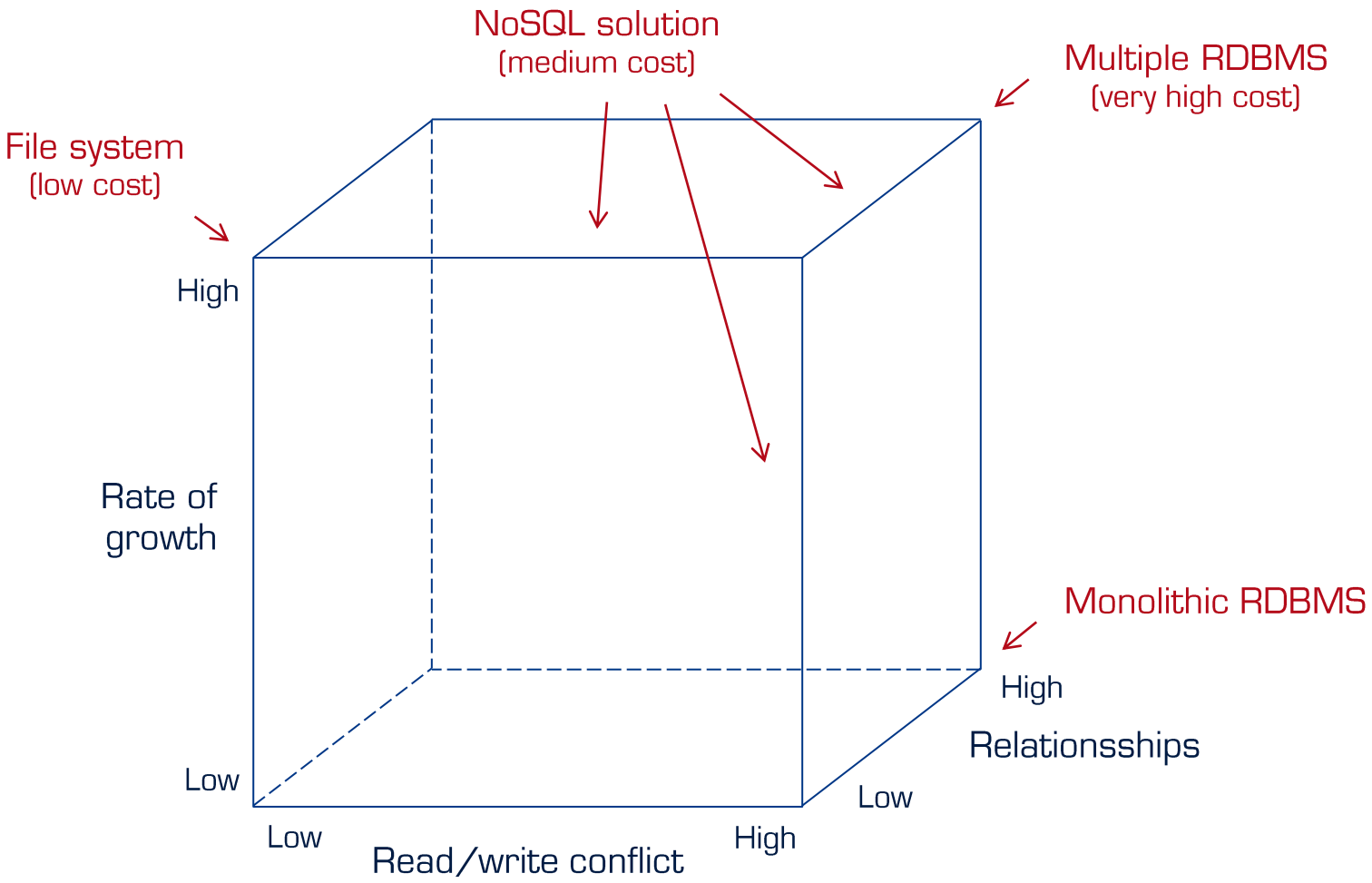
NoSQL

- Fills the gap between file systems and RDBMS
- Very diverse technologies: KV, Wide column, Document, Graph
- Different strategies w.r.t. CAP and scalability support
- Searching is often an issue
- Relatively young technology (stability, tools, documentation)

RDBMS

- Very strong at relations and concurrency handling
- Don't scale well beyond a certain boundary
- Mature technology, very good tool support

STORAGE TECHNOLOGY DECISION CUBE

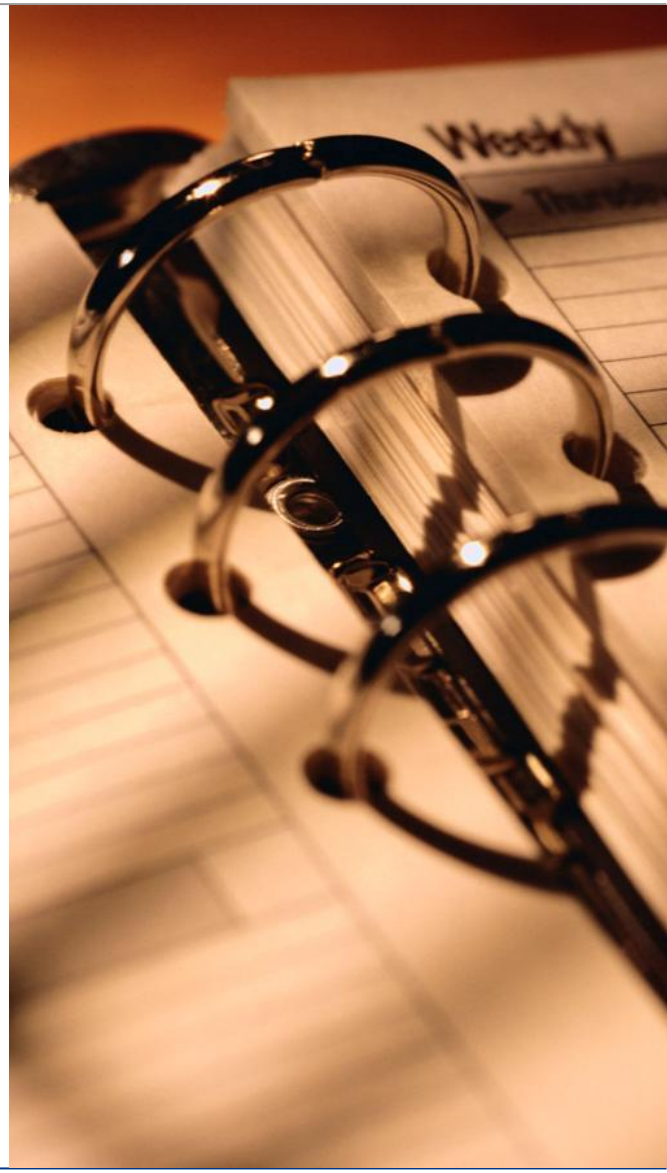


Source: [1]

Be wary of vendor solutions

AGENDA

- Basics of scalability
- Dimensions of scalability
- Design for high scalability
- Choose the right tool
- More stuff …
- Summary



MORE PATTERNS AND PRINCIPLES …

Fault tolerance

- Be prepared for things to crash
- Deal with inconsistencies

Database handling

- Use the right type of locking
- Don‘t use ‘*‘ in select statements

Using web technologies best

- Unleash the potential of HTTP

Extreme scalability

- Automation and elasticity
- Big disjoint entities with message based communication

And many more …

- Monitoring the system
- Designing and learning recommendations
- Never down system

MORE TO READ …

[1] Michael T. Fisher, Martin L. Abbott,
Scalability Rules: 50 Principles for Scaling Web Sites,
Addison-Wesley Longman, 2011

[2] Theo Schlossnagle, Scalable Internet Architectures,
Sams, 2005

[3] Jinesh Varia, Architecting for the Cloud: Best Practices,
Amazon Web Services 2010

[4] Jinesh Varia, Cloud Architectures,
Amazon Web Services 2008

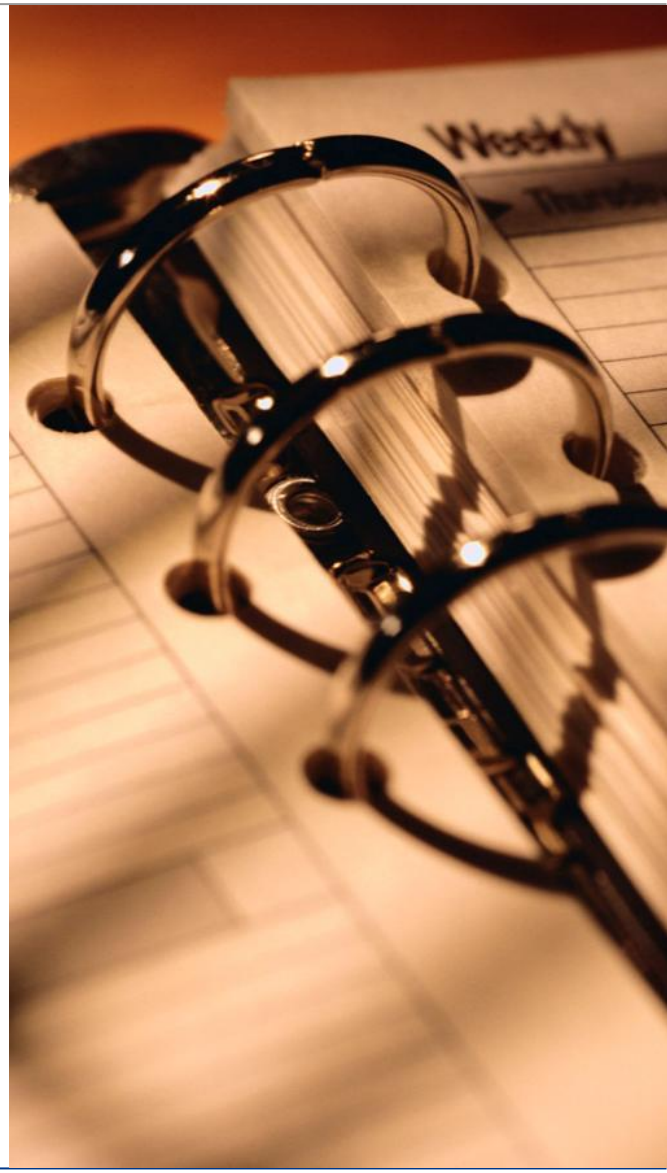
[5] Pat Helland, Life beyond Distributed Transactions,
3rd Conference on Innovative DataSystems Research
(CIDR) 2007

[6] http://highscalability.com/

[7] http://www.slideshare.net/jboner/
scalability-availability-stability-patterns

AGENDA

- Basics of scalability
- Dimensions of scalability
- Design for high scalability
- Choose the right tool
- More stuff …
- Summary



Scalability is primarily a design issue, not a tool or product issue

The core principles of scalability: simplicity, scale out, share nothing, asynchronous communication

The three dimensions of scaling

The different storage choices

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!

Uwe Friedrichsen
CTO

codecentric AG
Merscheider Straße 1
42699 Solingen

uwe.friedrichsen@codecentric.de
tel +49 (0) 212 . 23 36 28 10
fax +49 (0) 212 . 23 36 28 79
mobil +49 (0) 160 . 90 62 66 00

www.codecentric.de
blog.codecentric.de
www.meettheexperts.de

DISKUSSION & FRAGEN