Persönlichkeiten bei bluehands

bluehands







Technologien bei bluehands

bluehands

Microsoft Partner
Gold Independent Software Vendor (ISV)

Think Big

Skalierbare Anwendungen mit Windows Azure

Aydin Mir Mohammadi

bluehands GmbH & co.mmunication KG

am@bluehands.de; posts.bluehands.de/am

Immer mehr…

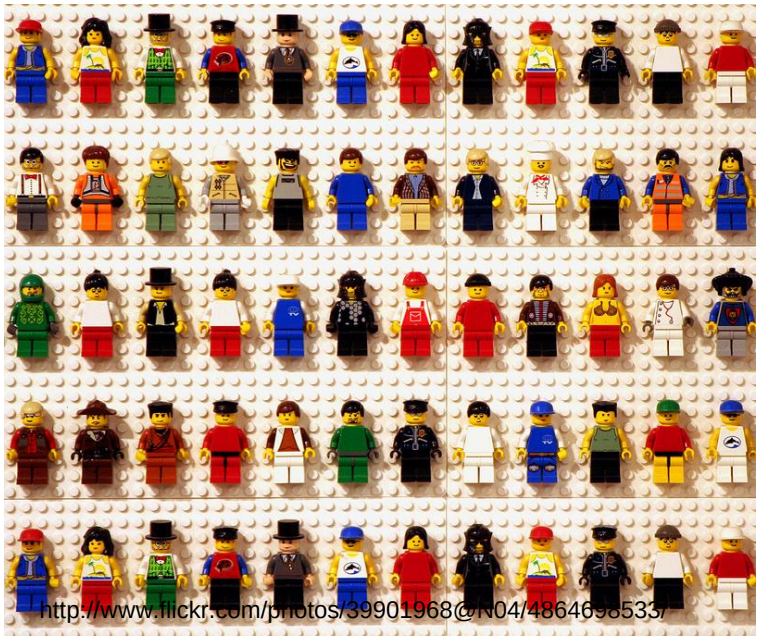
bluehands

- Mehr Performance
- Mehr Menge
- Mehr Verfügbarkeit

Skalierung

bluehands

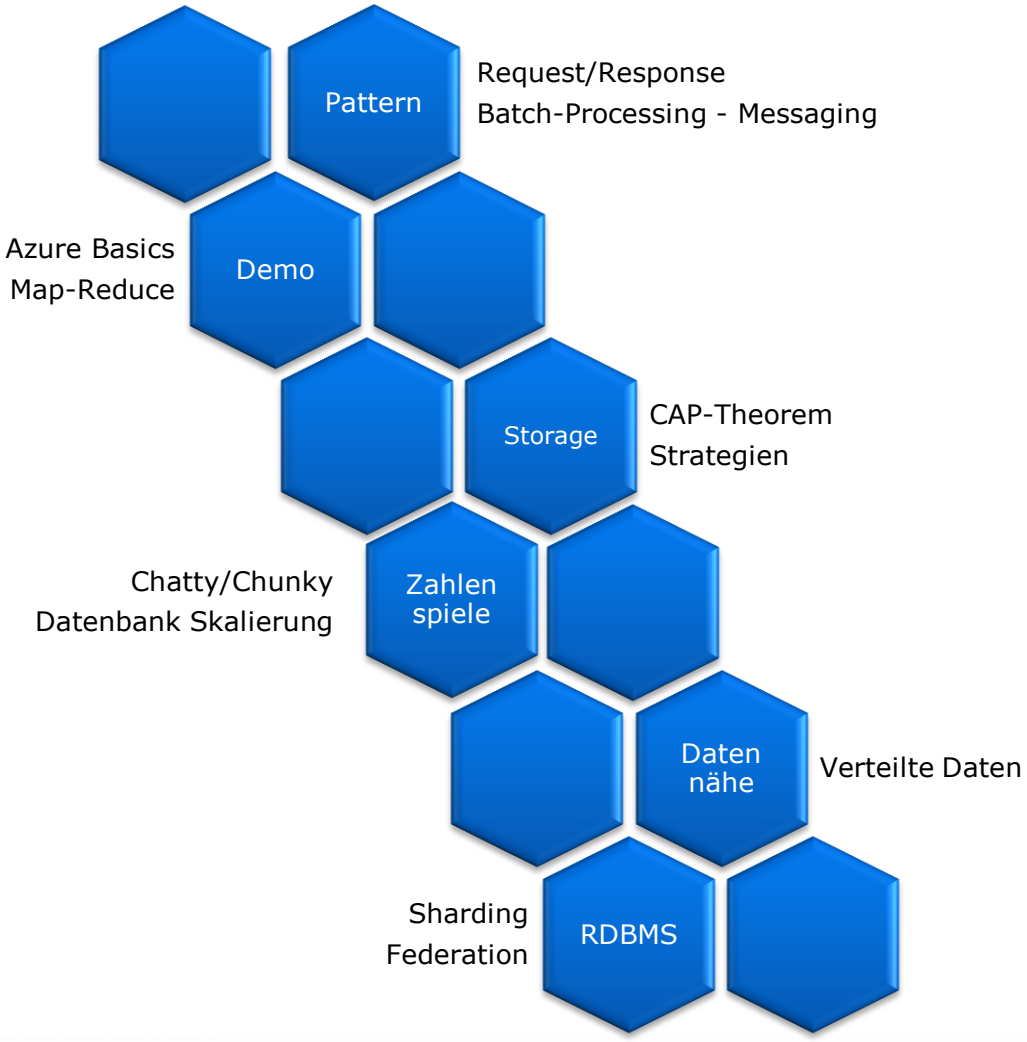
Vertikale Skalierung



Horizontale Skalierung

Agenda

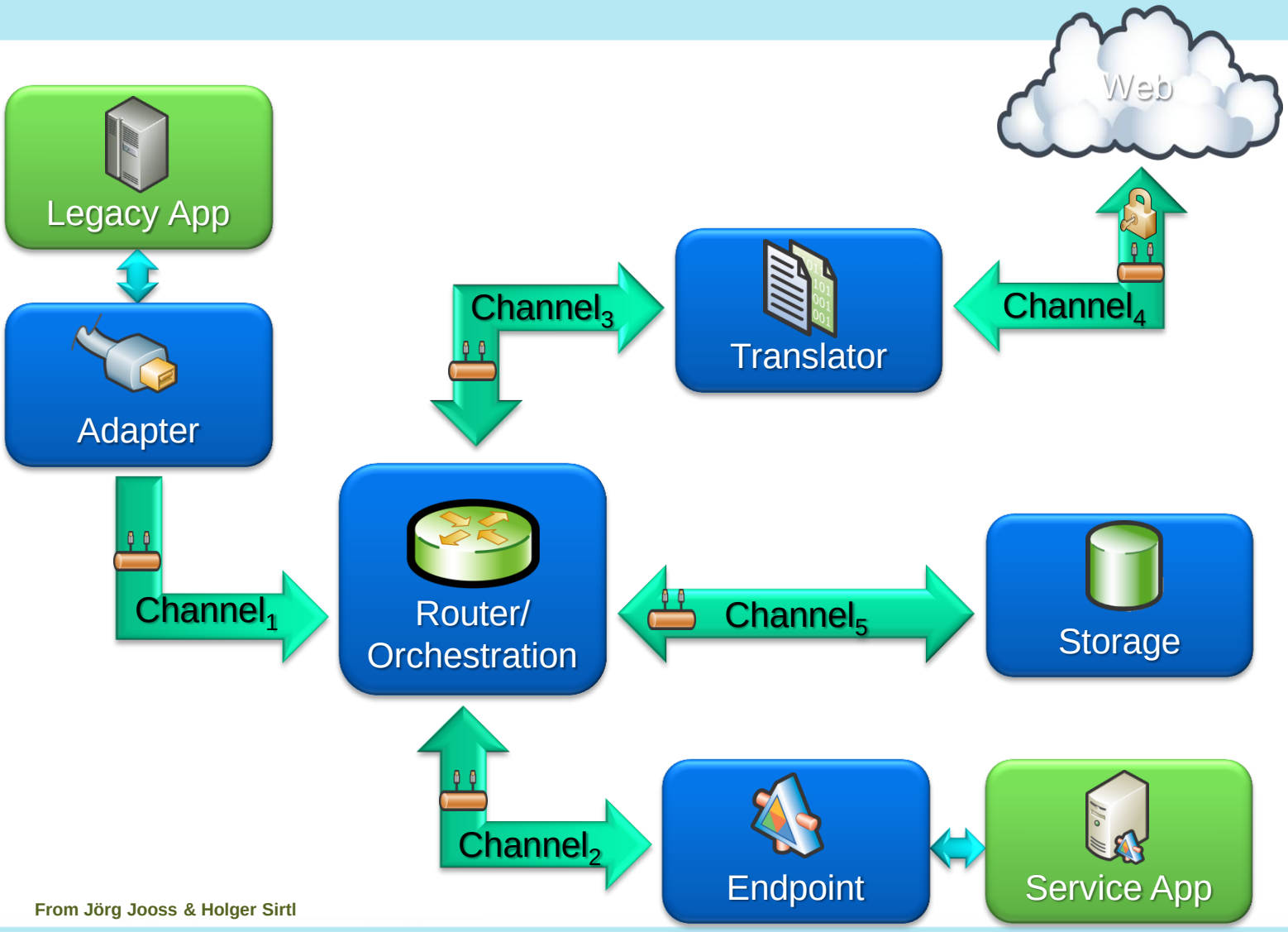
bluehands



Pattern

Pipes & Filter

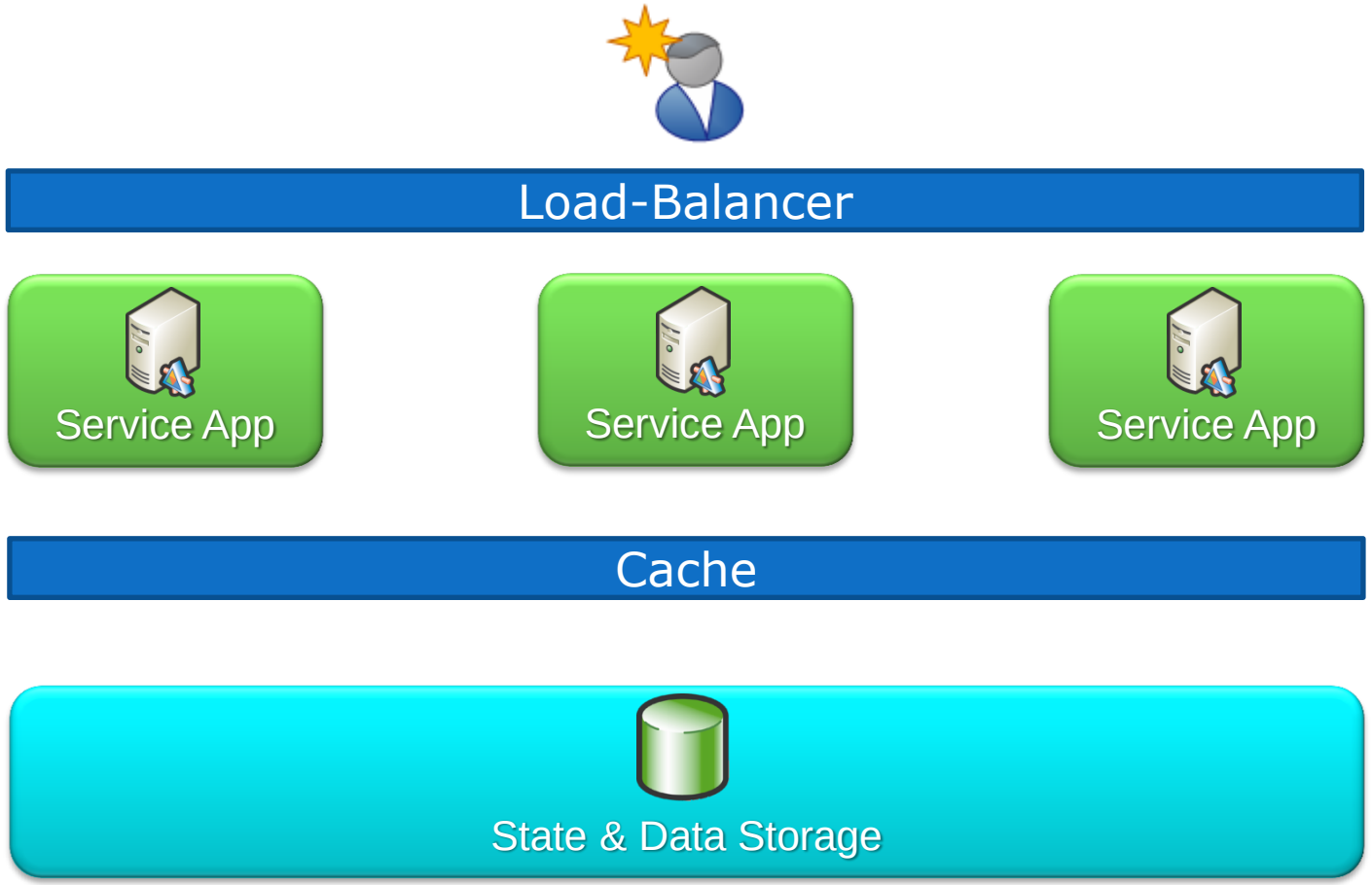
bluehands



From Jörg Jooss & Holger Sirtl

Pattern: Request/Response

bluehands



Pattern: Request/Response

bluehands

- Absoluter Klassiker
- Application-State am besten beim Client
  - Alternative: Session-Server (Cache)
  - Alternative: Instanz nicht balancen
- Daten im Cache, sonst Storage

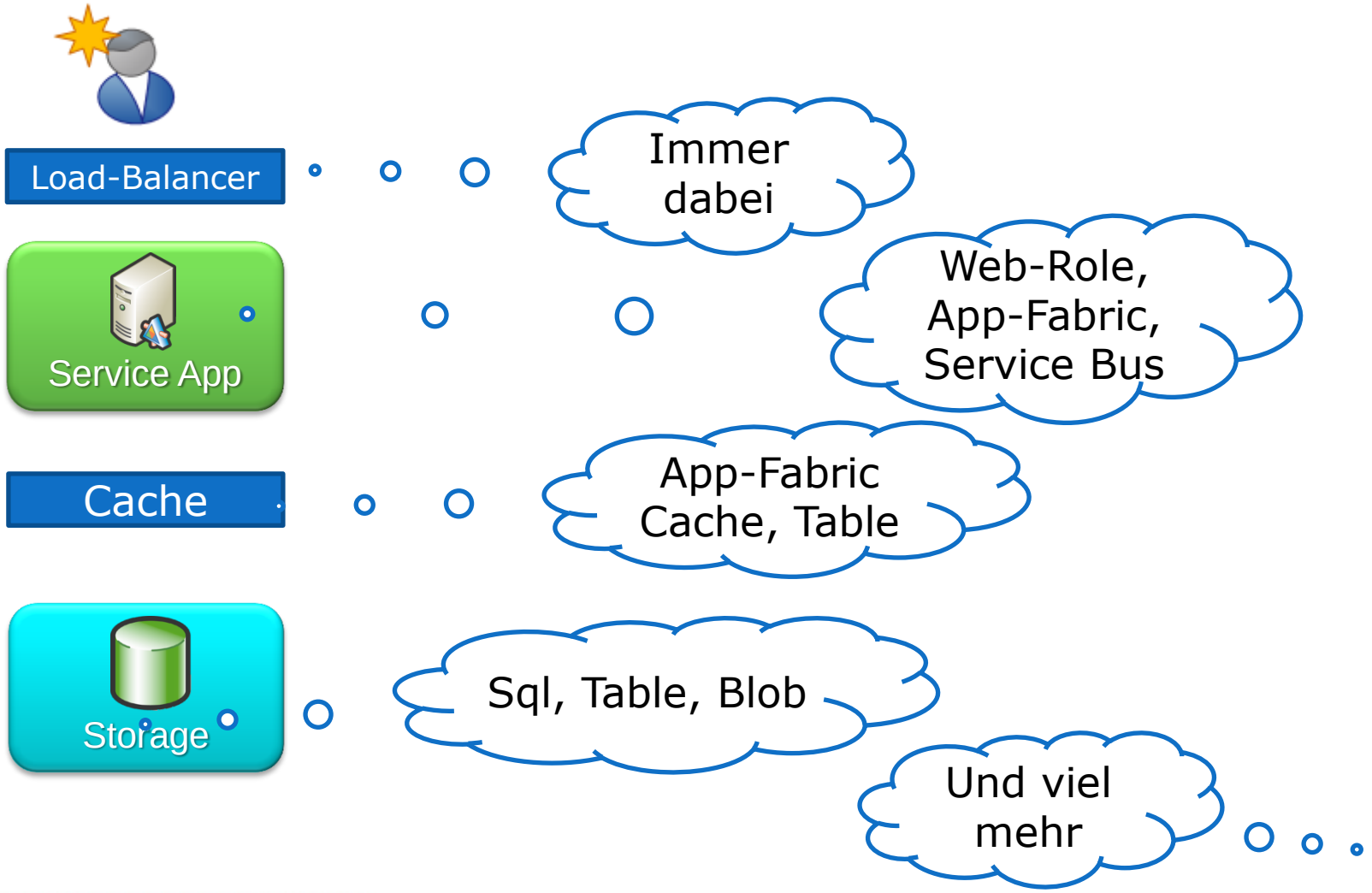
Pattern: Request/Response

bluehands

- Skalierung wird verlagert auf
  - Cache
  - Storage
- Größtes Problem
  - Cache Invalidisierung

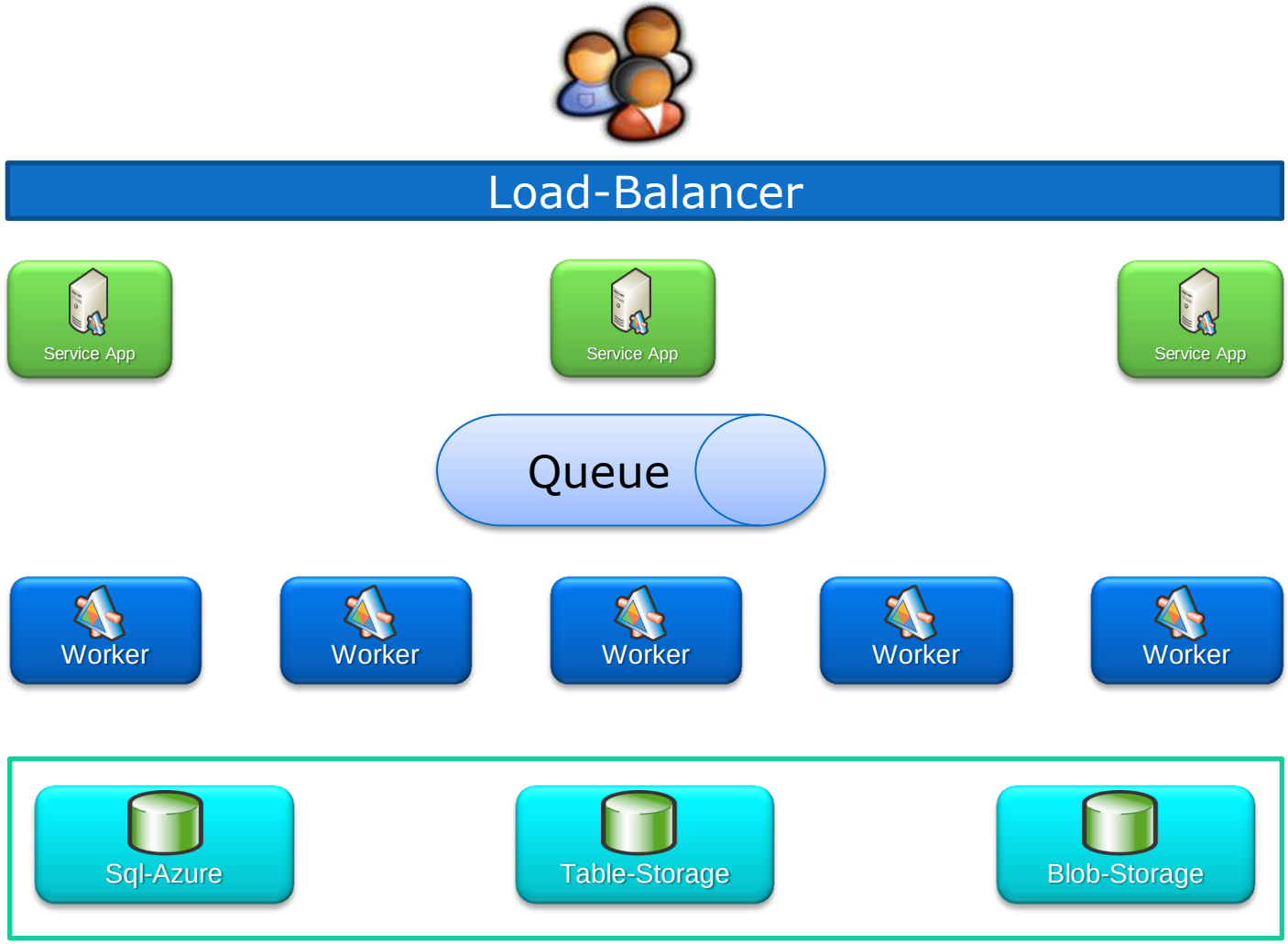
Pattern: Request/Response

bluehands



Pattern: Messaging

bluehands



Pattern: Messaging

bluehands

- Absoluter Klassiker
- Beste Skalierung
- Kann mit Request/Response kombiniert werden
- Auch hier gilt: Skalierung hängt am Storage

Demo

Azur Basics

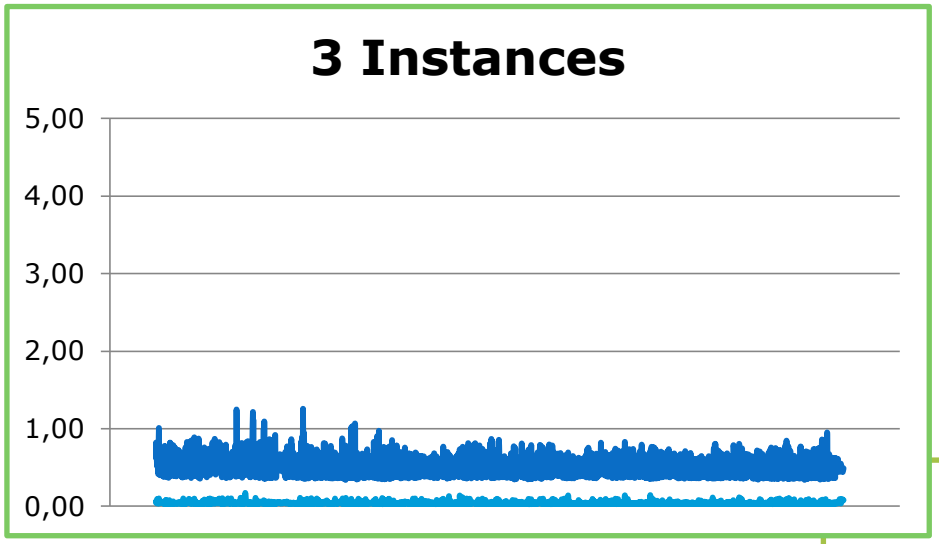
Binsenweisheit

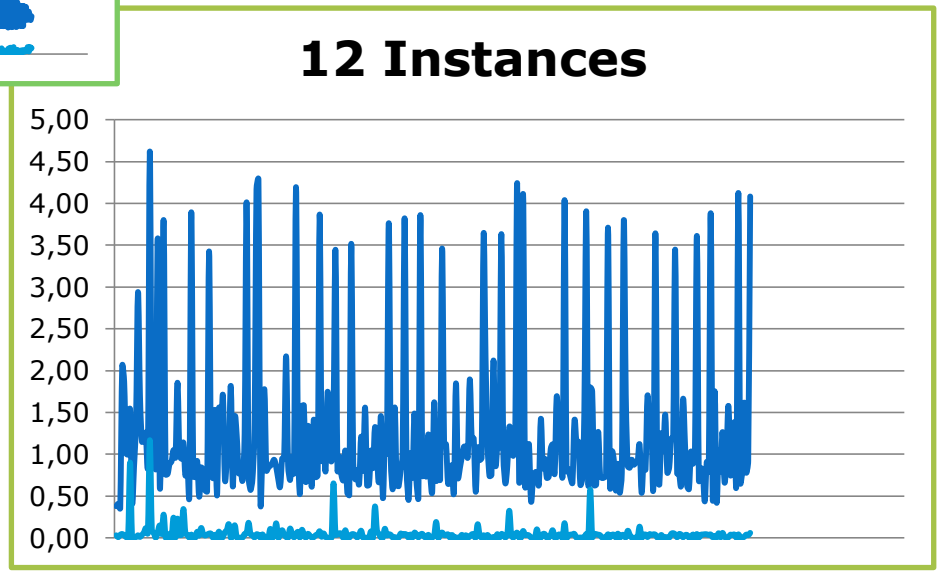
bluehands

- Am Ende hängt es an den Daten
  - Klassische Datenbanken skalieren nicht

RDBMS: Performance no. clients

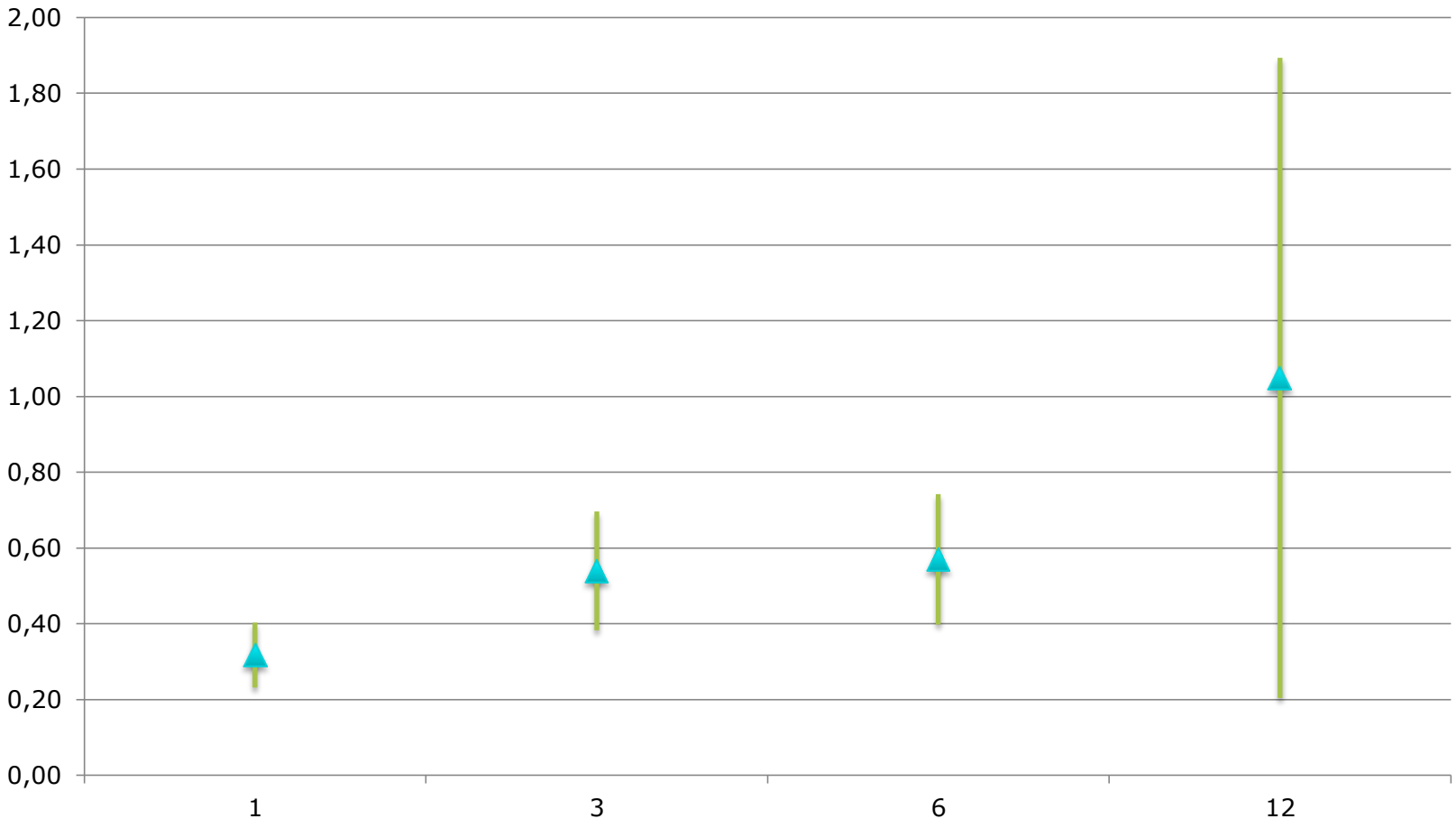
bluehands





RDBMS: Performance no. clients

bluehands



Binsenweisheit

bluehands

- Man muss die Daten „verteilen“
  - Willkommen in der Hölle

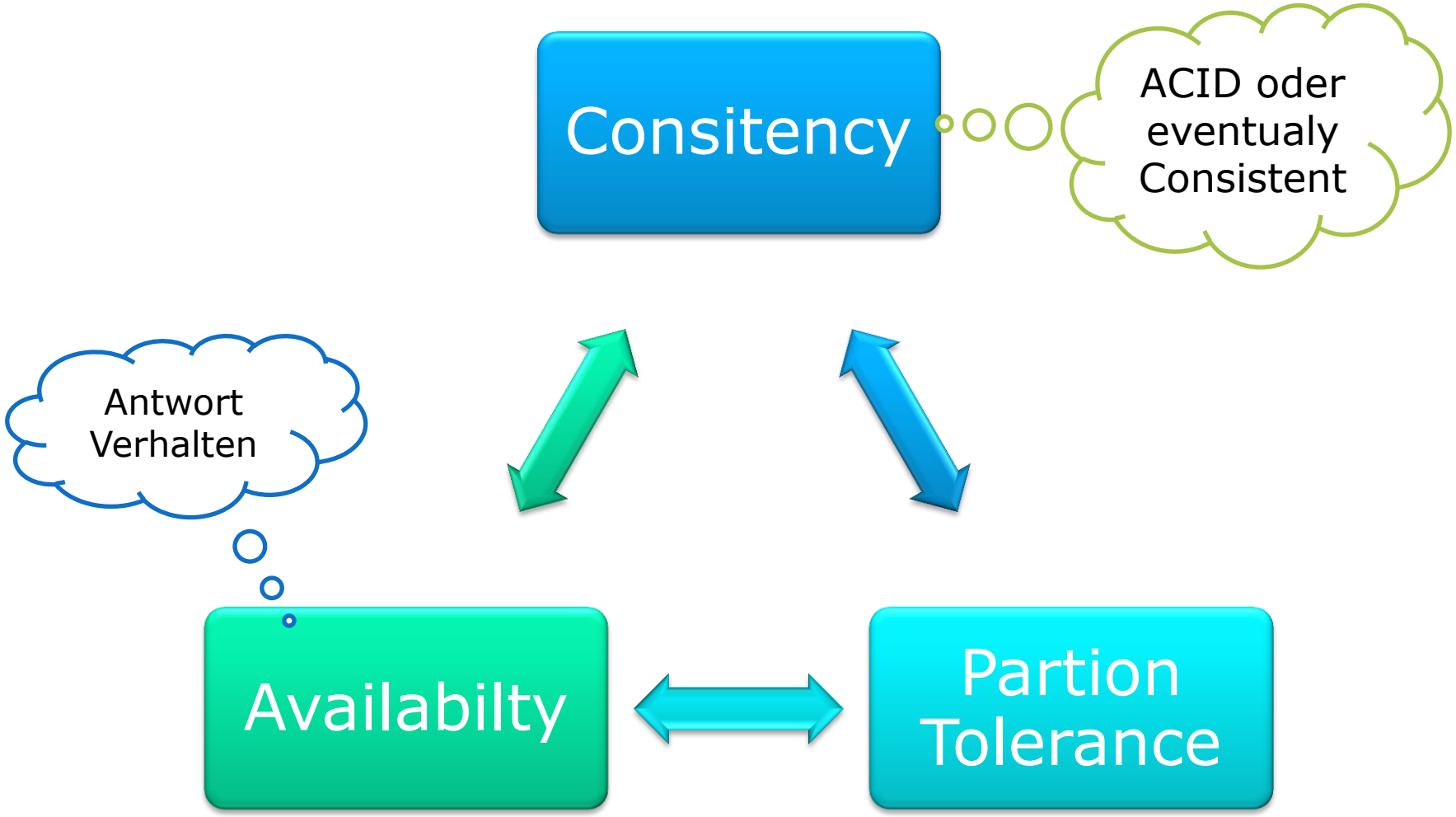
CAP-Theorem

bluehands

- Betrifft ein System mit verteilten Daten.
- Betrachtet folgende Eigenschaften
  - Consistency: Alle sehen gleichzeitig das gleiche
  - Availability: Alle können lesen & schreiben
  - Partition Tolerance: Ausfall eines Knotens führt nicht zum Ausfall des Systems

CAP-Theorem: Wähle 2 aus 3

bluehands



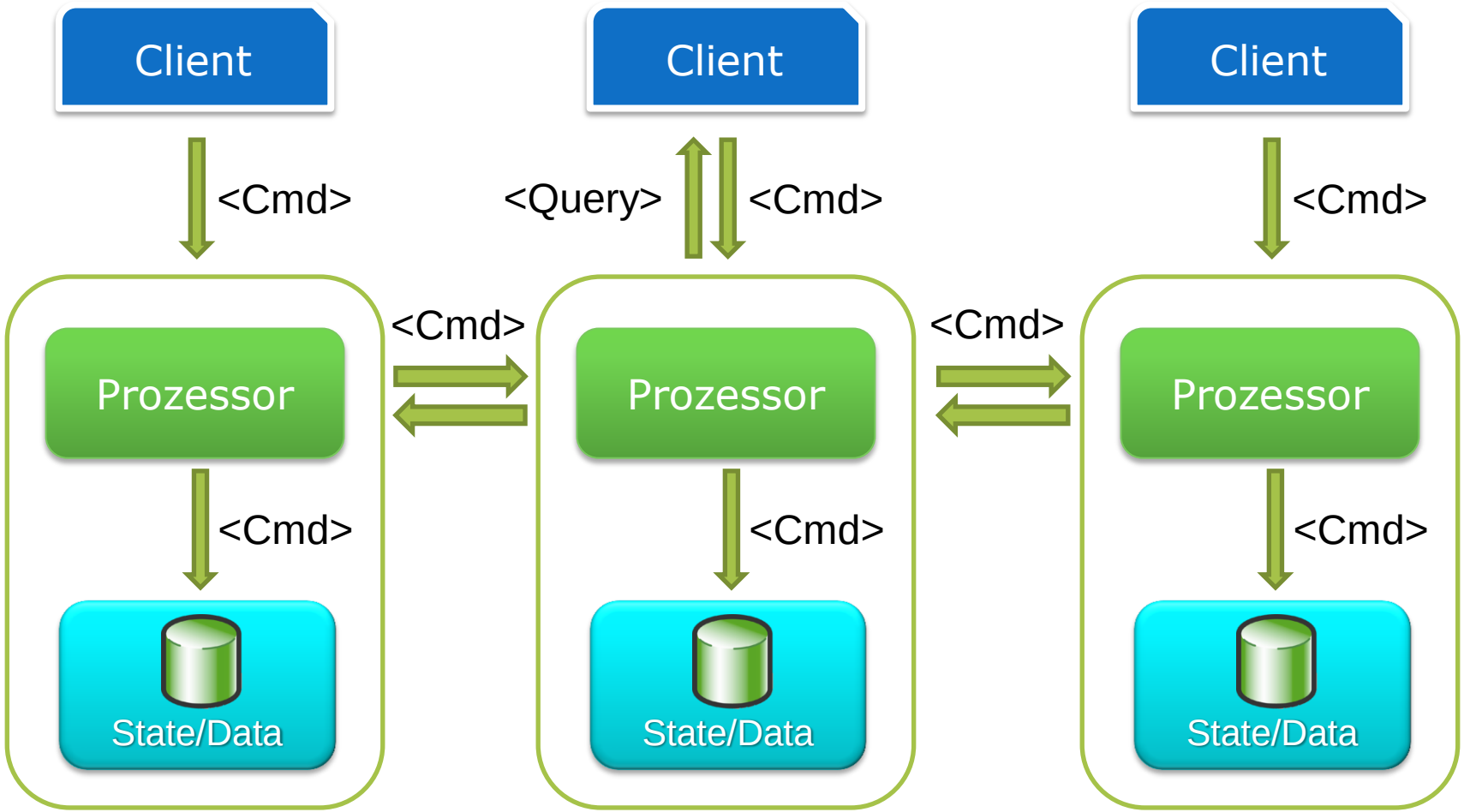
CAP-Theorem: Beispiele

bluehands

- Oracle Cluster (RAC)
  - Kein CAP-Theorem, da nicht verteilt
- Datenbank Mirroring (Log-Shipping)
  - Synchrone Commits: CA
  - Asynchrone Commits & Sync: AP
- Partitionierung
  - Sharding & Federation: CA

State & Pub/Sub Server: Async Commit

bluehands

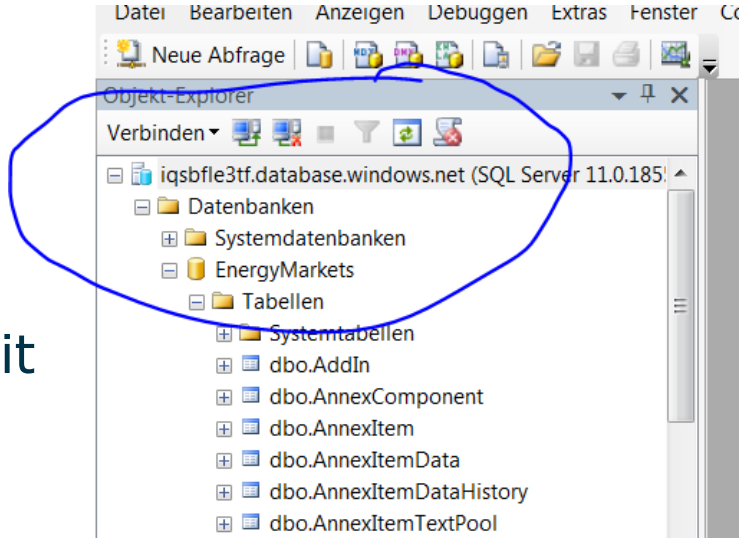


Sql Azure

RDBMS: Sql-Azure

bluehands

- Sql-Server in Azure
- Immer drei Knoten
  - 1 Primary, 2 Standby
  - 1 Sync commit, 1 async commit
- Funktional eingeschränkt
  - Kein Xml
  - Kein CLR
  - Kein OleDB (not supported)
- It just works



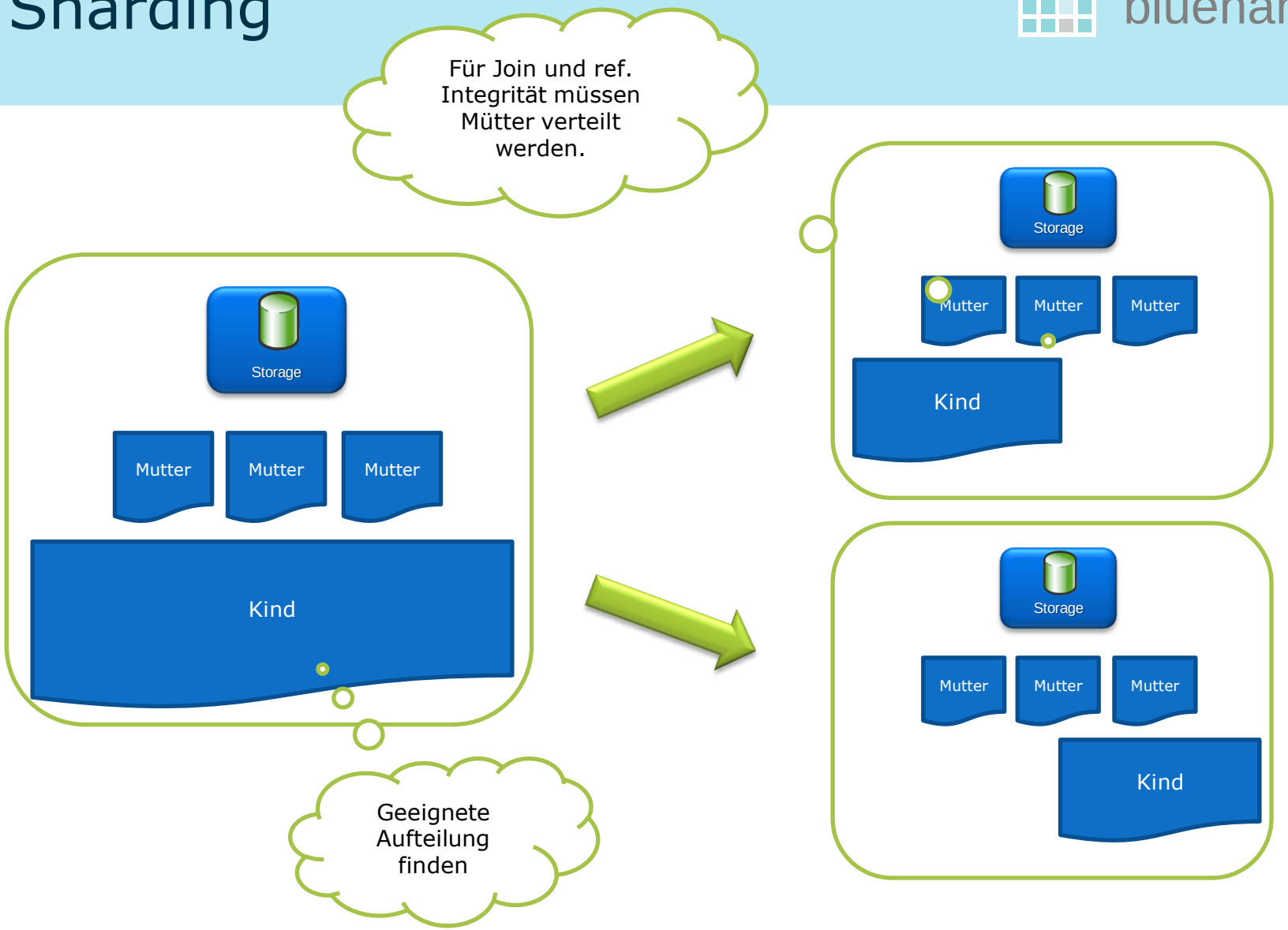
DBMS: Sql-Azure

bluehands

- Latenz beachten
- Licht braucht 1,8 ms
- Sql Ping ca. 15 ms

Sharding

bluehands



Sql-Azure: Skalierung (CA)

bluehands

- Manuelles Sharding
  - Daten werden auf mehrere Knoten verteilt. Zugriff wird im Client gesteuert.
- Sql Azure Federation
  - Eingebautes Sharding. Zugriff wird im Server gesteuert, jedoch nicht transparent.

Federation

bluehands

- Limits
  - Keine Transaktionen über shards. Verteilte Transaktionen sind in Sql-Azure nicht supported
  - Kein Auto-Increment
- Vorteile gegenüber manuelles Sharding
  - Online Split, Management
  - Datenbank kennt die Verteilung. Shard kann abgefragt werden

Federation

bluehands

- Federation „key“ überlegen
  - Über welche Eigenschaft einer Tabelle wird verteilt
- Federations erstellen
  - Man kann die einzelnen Shards immer wieder splitten
- Sql-Statements anpassen
  - Vor jedem Statement: Use Federation xxx (key=value),with filtering=off, reset

Table Storage

Table-Storage

bluehands

- Schemalos.
- Partitioniert. Jede Partition kann auf eine andere Maschine gehalten werden.
- Jede Entität hat ein PartitionKey und ein RowKey
- Versionierung über Timestamp

Table-Storage: How to use

bluehands

- Queries nur auf RowKey und PartionKey mit „=„
  - Table-Storage ist eine Hash-Table.
  - Alle andere Operationen machen einen Full-Scan.
  - Evtl. PartitionKey und danach Properties
- Limits beachten
  - 64k pro Eigenschaft, 1 MB pro Entität
  - 5000/sec auf Account und 500/sec auf Partition

Table-Storage: How to use

bluehands

- Komplizierte denormalisierte Objekte
  - Z.B.: Profile, Settings
  - Eher statische Entitäten
- Vorberechnete Sichten. Daten werden je nach Anwendung vorbereitet.
  - In RDBMS: Mehrere Clustered Indizes

Blob Stroage

Blob-Storage

bluehands

- „Filesystem“ in der Cloud
- Content ist von überall abrufbar
- Über http als Ressource einbinden

Query

Map-Reduce

bluehands

- Idee
  - Query in kleine Teile aufteilen
  - Auf viele Knoten ausführen
- Voraussetzung
  - Code und Daten sind nah
  - D.h. Map-Reduce auf einer zentralen DB macht in der Regel keinen Sinn

Map-Reduce

bluehands

- In Linq
  - map --> Enumerable.Select
  - reduce --> Enumerable.Aggregate
- Mehrere Implementierungen (deprecated)
  - DryadLINQ
  - Daytona
- Hadoop

Kosten

Kosten

bluehands

- Nicht geschenkt!
  - 3 Dienste => 6 kleine Server (510,63$)
  - 1 DB =>150 GB (160,12$)
  - Speicher => 1.000 GB & Transkation (159,58$)
  - Netzwerk => 2.000 GB (170,21$)
  - Cache, Authentifizierung (~100$)
  - Summe: 1.100$/Monat
- Deutlich billiger als interne IT
- Mini Website: 16$/Monat
- Table & Blob Storage deutlich billiger als Sql Azure

Fazit

bluehands

- Skalierung wird einfacher
  - Gleich mit designen
- Infrastruktur wird besser
  - Datenbanken gehen mit
- Think Big

Fragen?

Links

bluehands

- http://things.smarx.com/
- http://azuresamples.com/
- http://azurestoragesamples.codeplex.com/
- http://research.microsoft.com/en-us/projects/daytona/default.aspx
- http://blogs.msdn.com/b/windowsazurestorage/archive/2010/11/06/how-to-get-most-out-of-windows-azure-tables.aspx
- http://social.technet.microsoft.com/wiki/contents/articles/2281.federations-building-scalable-elastic-and-multi-tenant-database-solutions-with-sql-azure.aspx
- http://blogs.msdn.com/b/cbiyikoglu/
- http://developer.yahoo.com/hadoop/tutorial/index.html