

Erfahrungen mit MongoDB für die Verwaltung meteorologischer Massendaten

Richard Lutz, Institut für Angewandte Informatik (IAI)

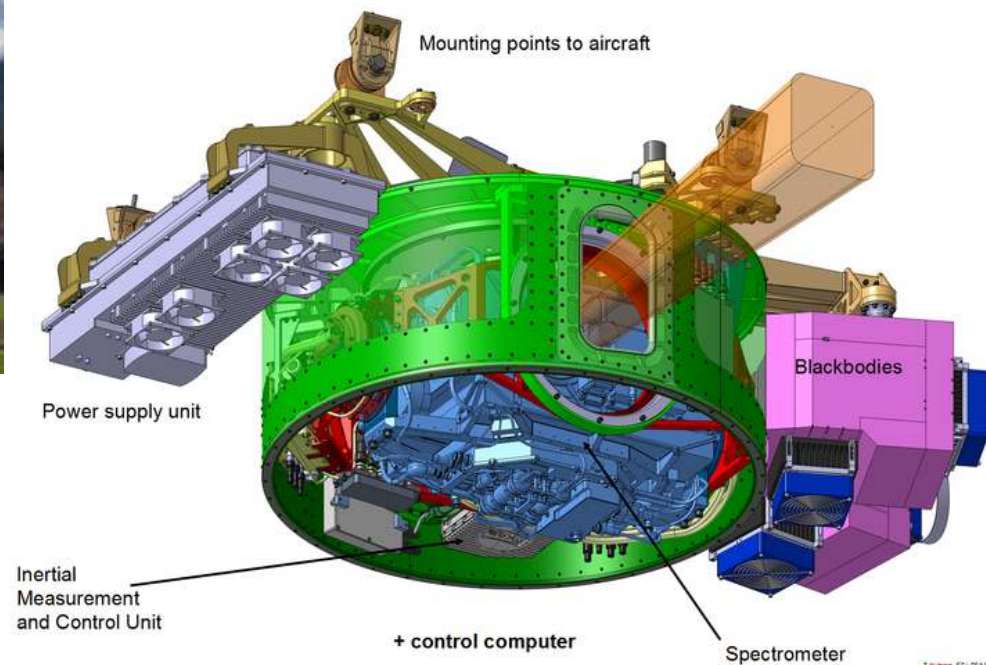


Inhalt

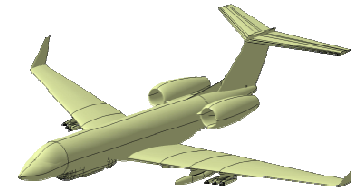
- Das Messsystem HALO/GLORIA
- Struktur der GLORIA-Daten
- Anforderungen an die GLORIA-Datenbank
- System- und Testumfeld
- MongoDB
 - Externe Struktur (Dateisystem)
 - Interne Struktur (DB, Collection, Document)
 - mongo-Shell
 - Java-Interface
- Erfahrungen mit MongoDB
- Zusammenfassung

Das Messsystem HALO/GLORIA

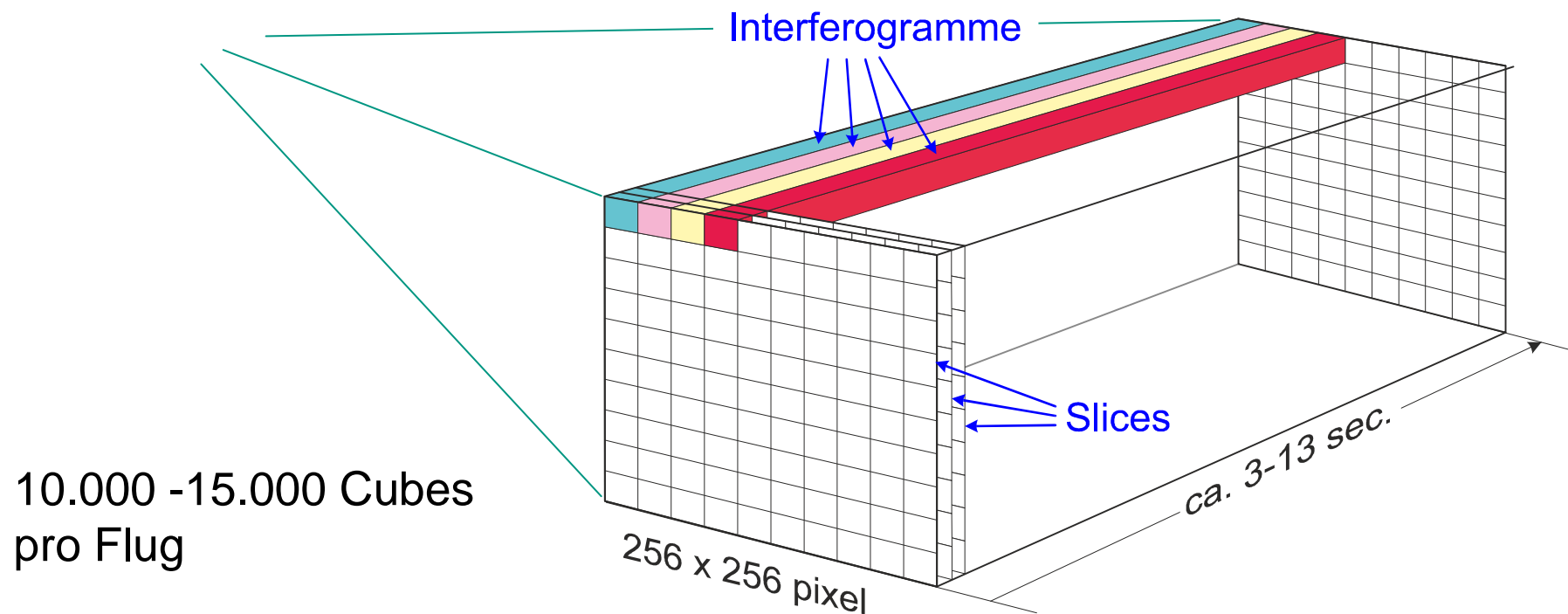
- GLORIA - Gimballed Limb Observer for Radiance Imaging of the Atmosphere (misst Infrarotstrahlung in der Stratosphäre)
- HALO - High Altitude and Long Range Research Aircraft
- Mehrere Kampagnen in 2012



Struktur der GLORIA-Daten



- Speicherung der Messaufnahmen in so genannten **Cubes**



Anforderungen an die GLORIA-Datenbank

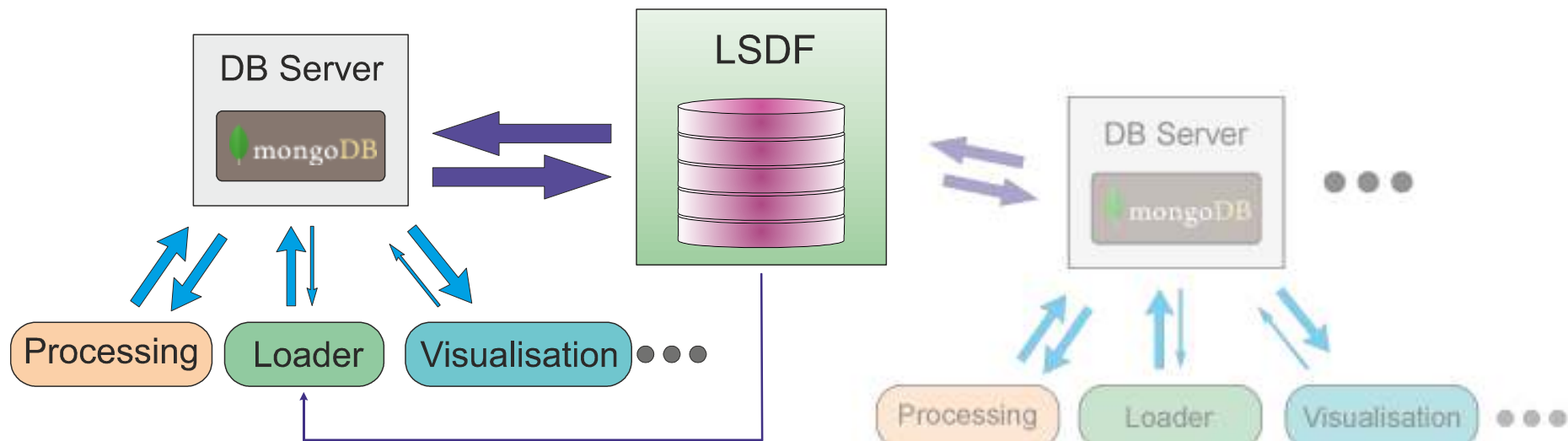
- Zugriff auf Daten und Metadaten mit einer einzigen Abfrage
- Frei verfügbare Datenbank mit einfacher Verwaltung
- Leichte Erweiterbarkeit der Daten für unterschiedliche Datentypen
- Schneller Zugriff auf große Datenmengen
 - Ausgewählter Flug: Flug 08 vom 28.08.2012
 - ≈ 2,1 TB
 - ≈ 10.200 Cubes
 - ≈ 62 Mio. Interferogramme bzw.
 - ≈ 162 Mio. Slices
- Verfügbarkeit leistungsfähiger Schnittstellen für Java, Python, C++ usw.
- Möglichkeit der Speicherung von Daten in Arrays

MongoDB (→ www.mongodb.org)

- NoSQL-Datenbank
- Schemafrei (JSON Data Model mit Dynamischen Schemata)
- Frei verfügbar (Open Source, in C++ geschrieben)
- Für große Datenmengen ausgelegt
- Spezielle Dateien für große Datenblöcke (GridFS)
- Auto-Sharding (Horizontal Scalability)
- Built-In Replication (High Availability)
- Kommandozeilen-Tool
- Zahlreiche Tools verfügbar im Open-Source-Bereich
- Aggregation Framework & Native MapReduce
- Hadoop-Integration
- 50 Treiber (10 MongoDB-supported, 40 Community-supported)
- Sehr ausführliche und sehr gute Online-Dokumentation

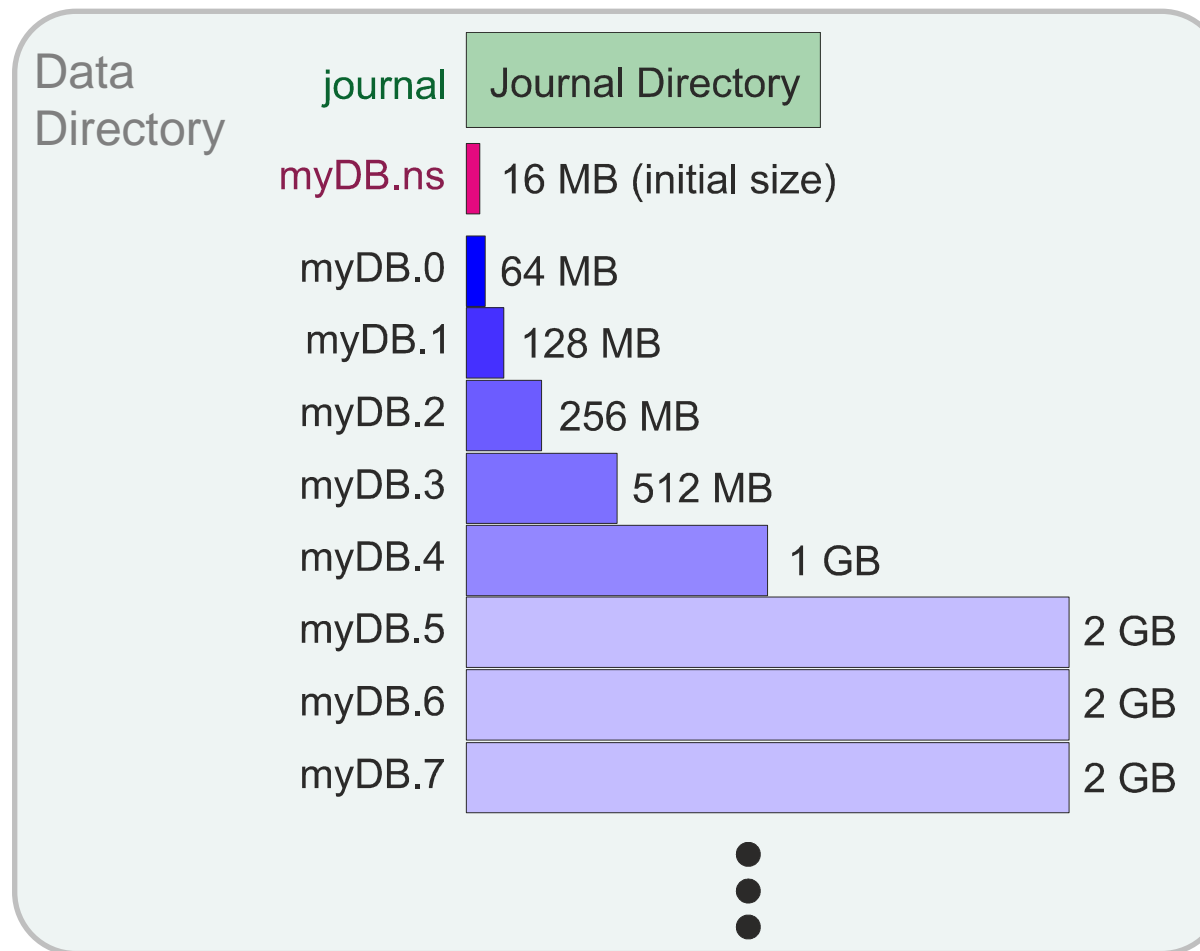
Systemumfeld der GLORIA-Datenauswertung

- Hardware: Intel I7,12 CPUs, 128 GB RAM, 64-bit-Linux (CentOS 6)
- MongoDB-Server: Version 2.4.8
- Speicherung: **LSDF** (Large Scale Data Facility) mit 10G Ethernet
- *Sharding* und *Replication* wird zurzeit untersucht



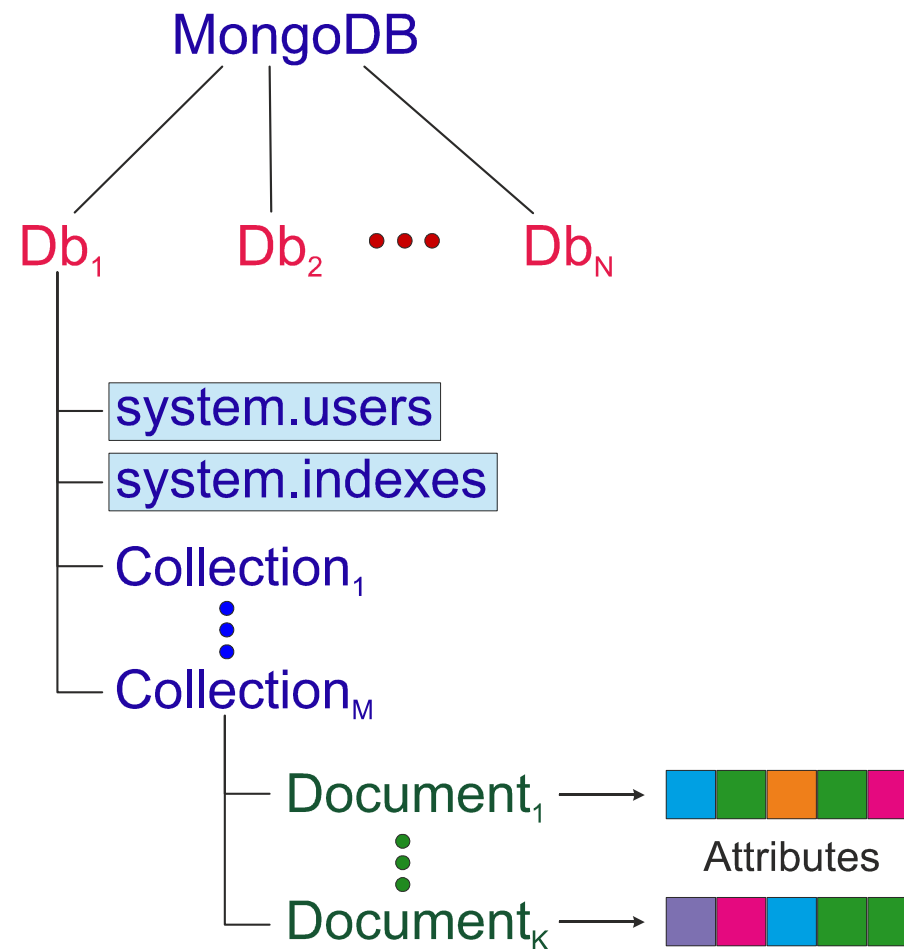
Dateistruktur von MongoDB

- MongoDB-Dateistruktur für die Datenbankinstanz *myDB*



Interne Struktur von MongoDB

- Datenbank-Instanzen, Collections, Dokumente und Attribute



Arbeiten mit MongoDB – die *mongo*-Shell

- Teil der MongoDB-Distribution
- Stellt eine vollständige JavaScript-Umgebung zur Verfügung
- Vollständige Datenbank-Schnittstelle für MongoDB

Beispiele:

- `db.myColl.insert({ attr1: "any text", attr2: 1000, abcChars: ["a", "b", "c"] })`
- `j = { name : "mongo" }`
- `k = { x : 3, defChars: ["d", "e", "f"] }`
- `db.myColl.insert(j)`
- `db.myColl.insert(k)`
- `var c = db.myColl.find({ attr1: "any text", attr2: { $gt: 900 } }).sort({ attr2: 1 })`
- `while(c.hasNext()) printjson(c.next())`

Arbeiten mit MongoDB – Java-Schnittstelle

- MongoDB-Elemente sind durch entsprechende Klassen abgebildet
- DB-Objekte sind BSON-Objekte (HashMaps mit Key/Value-Paaren)
- Funktionalität ist JavaScript bzw. *mongo*-Shell nachempfunden

Beispiel:

```
MongoClient mongoClient = new MongoClient(  
    new MongoClientURI( "mongodb://" + user + ":" + pwd + "@" + host + ":" + 27017 ) );
```

```
DB db = mongoClient.getDB( "myDB" );  
DBCollection myColl = db.getCollection( "myColl" );
```

```
DBObject myObject = new BasicDBObject( "attr1", "any text" ).append( "attr2", "1000" );  
String[] abcChars = new String[]{ "a", "b", "c" };  
myObject.add( "abcChars", abcChars );  
myColl.insert( myObject );
```

```
DBCursor curs = myColl.find( new BasicDBObject( "attr1", "any text" ) );  
// komplexe Abfragen: DBObject dbo = com.mongodb.util.JSON.parse( String jsonQuery )
```

Erfahrungen mit MongoDB

- Zugriff auf Daten und Metadaten mit einer einzigen Abfrage
 - Leicht möglich, da Daten **in** der Datenbank liegen (nicht nur referenziert)
- Installation und Handhabung
 - Sehr einfach. Sehr gute Dokumentation
- BSON-Datentypen und Konvertierungen
 - MongoDB: Daten in BSON-Format (Binary JavaScript Object Notation)
 - GLORIA-Daten überwiegend in 16-Bit-Short-Werten
 - „kleinstes“ MongoDB-Element ist 32-Bit-Integer
 - ➔ Verdoppelung der Datenmenge
 - ➔ Konvertierung und Speicherung der Daten in Byte-Arrays

Erfahrungen mit MongoDB (2)

■ Indizierung und Datenzugriff

- leistungsfähiger Index- und Caching-Mechanismus
- Kombinierte Indizes möglich, können auch partiell verwendet werden
- Alle Attribute, nach denen gesucht wird, sollten unbedingt indiziert sein
→ sonst sehr langsam
- Beispiele:
 - Suche nach best. Slice-Index über alle Cubes des Fluges: $\approx 8s$
 - Suche nach allen Interferogrammen eines Cubes: $\approx 1s$
 - Suche nach best. Interferogramm-Index über alle Cubes: $\approx 40s$
- Abfrage stark abhängig von Caching (Wiederholung der Abfrage)
- Ergebnis der Suche: Referenzen, Download exklusive

■ Laden und Lesen von Daten

- Hängt stark vom Caching ab
- Hängt auch von Reihenfolge der Speicherung (Anordnung) ab

Zusammenfassung

- MongoDB sehr gut geeignet zur Erfüllung der Anforderungen
- Vorteile:
 - Freie Open-Source-Software
 - einfache und schnelle Installation
 - einfaches Management der Benutzerkonten und Inhalte.
 - Schneller Zugriff auf die Daten
 - Lesen der Attribute und des kompletten Datensatzes mit einziger Abfrage
 - Alle benötigten Programmierschnittstellen verfügbar
 - Sehr leistungsfähiger Indizierungs- und Caching-Mechanismus
- Nachteile:
 - Fehlen primitiver BSON-Typen wie *short*, *float*, *int*
 - Suche mit nicht indizierten Attributen sehr langsam

Vielen Dank für Ihre Aufmerksamkeit!