# Thrift, Protocol Buffers, Avro. Welches Datenformat?

== Normale Anwendungen ==
Für "normale" Anwendungen haben sich Formate wie XML oder JSON etabliert.
XML ist der quasi Standard wenn es um einen strukturierten Datenaustausch geht. Es ist selbst als Computer Endanwender schön lesbar, man kann XML in den meisten Programmiersprachen nutzen ohne irgendwelche Spezialsoftware installieren zu müssen. Mit Schema-Validatoren, Stylesheet, .. bildet es viele Anwendungsfälle ab.

Doch wenn es um Geschwindigkeit hat, hat XML den Nachteil, dass die Dateien sehr groß werden. Die eigentlichen Daten können gerne größer werden als das Format, das sie beschreibt, in dem Fall spricht von einem Protokolloverhead. Außerdem ist die Struktur sehr langsam zu verarbeiten.

Abhilfe sollen Binärprotokolle schaffen.

== Aufbau Binärprotokolle ==
Protocol Buffers und Thrift sind so aufgebaut, dass man ein Schema in einer "Interface Description Language" definiert und daraus dann mit Kommandozeilen Tools Models für die benötigten Programmiersprachen erstellen kann.

Damit man eines der folgenden Protokolle einsetzen kann, sollte man vorher schauen ob die Programmiersprache unterstützt wird. Die populärsten Programmiersprachen sind aber im allgemeinen ganz gut abgedeckt.

Durch den Support in vielen Programmiersprachen lassen sich die Protokolle sowohl im Client- als auch im Servercode nutzen.

== Vorstellung ==
=== Protocol Buffers ===easy-coding.de/Attachment/1111…b4f2d79e7f0eafba0cafd17ce
Protocol Buffers (Protobuf) wird bei Google zur Speicherung und zum Austausch strukturierter Daten verwendet und dient dort als Basis für ein RPC-System zur intermaschinellen Kommunikation. Es wurde 2001 eingeführt und 2008 als Open Source veröffentlicht.

Auch Twitter nutzt Protocol Buffers und hat unter dem Projektnamen "Elephant Bird" viele nützliche Tools beigesteuert die eine Nutzung von Protocol Buffers in verschiedenen Hadoop Produkten vereinfacht.

Protocol Buffers bringt Support für zahlreiche Programmiersprachen mit. Die Implementierung in Client- und Servercode bzw den Transport der Daten ist nicht Teil des Standard. Hier muss also entweder selbst implementiert oder auf 3rd Party Komponenten zurückgegriffen werden.

**Beispiel eines Schemas**

### Quellcode

```
1. message Point {
2. required int32 x = 1;
3. required int32 y = 2;
4. optional string label = 3;
5. }
6. 
7. message Line {
8. required Point start = 1;
9. required Point end = 2;
10. optional string label = 3;
11. }
12. 
13. message Polyline {
14. repeated Point point = 1;
15. optional string label = 2;
16. }
```

Alles anzeigen

=== Thrift ===easy-coding.de/Attachment/1112…b4f2d79e7f0eafba0cafd17ce
Thrift wurde von Facebook entwickelt und ist inzwischen ein Open Source Apache Projekt. Weil viele Ex-Googler daran gearbeitet haben, sagen viele dass es ein verbessertes Procol Buffers ist.

Thrift bringt einen RPC Stack mit. Erstellt man einen Code für seine Programmiersprache enthält man neben den Models auch einen Client- und Servercode für die Kommunikation.
Das kann aber auch Nachteile haben, Thrift ist stark mit den RPC Komponenten gekoppelt und es fällt nicht einfach seine Thrift Daten in simple Logfiles oder über Aggregatoren wie Flume und Scribe zu übertragen.

Thrift bringt komplexe Typen wie Listen und Maps mit.

**Beispiel eines Schemas**

### Quellcode

```
1. struct UserProfile {
2. 1: i32 uid,
3. 2: string name,
4. 3: string blurb
5. }
6. service UserStorage {
7. void store(1: UserProfile user),
8. UserProfile retrieve(1: i32 uid)
9. }
```

=== Apache AVRO ===easy-coding.de/Attachment/1113…b4f2d79e7f0eafba0cafd17ce
AVRO ist das jüngste Format. Es wurde 2009 entwickelt und schreibt sich auf die Fahne flexibler als Protocol Buffers und Thrift zu sein. Es nutzt JSON als zugrundeliegendes Datenformat und Client und Server tauschen ihr Format mit einer Art Handshake aus.

Durch die Flexibilität kann es in Zukunft direkt in Systemen wie PIG und HIVE genutzt werden.

Doug Cutting fasst die Vorteile von AVRO in einem Video zusammen: cloudera.com/blog/2009/11/avro…mat-for-data-interchange/

**Beispiel**

### Quellcode

1. {"type": "record", "name": "Point",
2. "fields": [
3. {"name": "x", "type": "int"},
4. {"name": "y", "type": "int"},
5. ]
6. }

== Zusammenfassung ==
Benchmarks zu den verschiedenen Protokollen findet man unter code.google.com/p/thrift-protobuf-compare/wiki/Benchmarking

== Beispielanwendung ==
Im Rahmen dieses Wiki Artikels wird noch ein komplexes Beispiel über das Zusammenspiel von Thrift, Protocol Buffers, Scribe, Flume, Hadoop und Pig erstellt.

== Literatur ==


- stackoverflow.com: Biggest differences of Thrift vs Protocol Buffers?