7BDIN006W
Big Data Theory and Practice

Lecture 7

Big Data and Data Transfer Formats

UNIVERSITY OF WESTMINSTER

Importance of Data Transfer Formats in Big Data

Role

Data transfer formats are the backbone of data interchange between services and systems. They define how data is packaged and communicated, impacting both the efficiency and the fidelity of data exchange.

Interoperability

These formats enable the *seamless flow of information across different technologies and platforms*, ensuring that data remains consistent and accessible regardless of the system in use.

Data Serialisation

Data serialisation is the process of converting data structures, or object states into a format that can be easily stored or transmitted and reconstructed later. The primary purpose of serialisation is to convert the complex data that's understandable by the program into a format that can be easily written to a file, database, or transmitted across a network.

When the serialised data reaches its destination, it is then *deserialised*—converted back into usable in-memory objects by the receiving program. This process is crucial for various programming tasks such as data storage, API integration, and communication between different parts of a distributed system.

For example, when you serialise an object to JSON format, you convert the object's properties into a string that can be easily written to a text file or sent over HTTP to a web server. Later, this string can be read and converted back into an object by the receiving system, which uses the same or compatible software to interpret the JSON format.

Serialisation formats:

- text-based (like JSON or XML) which are human-readable
- binary (like Avro or Protocol Buffers), which are more compact and faster to parse but not human-readable.

XML (eXtensible Markup Language)

XML data is hierarchical and tree-like, with parent and child elements.

Syntax: XML's syntax is tag-based, making it similar to HTML but designed specifically for data representation rather than displaying web content.

Schema: XML Schema Definition (XSD) provides a way to define the structure, content, and semantics of XML documents.

Namespaces: XML supports namespaces to prevent naming conflicts in XML documents when using tags that may have the same name but different meanings.

Pros and Cons

Pros: Human-readable and self-descriptive, XML is highly structured and widely supported across different systems.

Cons: The verbosity of XML leads to larger file sizes, and it can be slower to parse, especially for big datasets.

XML Data vs Tabular Representation

```
<books>
  <book>
    <title>Big Data for Dummies</title>
    <author>John Doe</author>
    <year>2021</year>
    <price>29.99</price>
  </book>

  <book>
    <title>Data Science 101</title>
    <author>Jane Smith</author>
    <year>2020</year>
    <price>35.95</price>
  </book>
</books>
```

Title	Author	Year	Price
Big Data for Dummies	John Doe	2021	29.99
Data Science 101	Jane Smith	2020	35.95

JSON (JavaScript Object Notation)

Data in JSON is organised in name/value pairs, with arrays and objects used for structure.

Syntax: JSON uses a text-based, lightweight format, which is easy to write and read for humans and to parse and generate for machines.

Schema: JSON Schema specifies rules for JSON data, such as what properties are required, their data type, and more.

Pros and Cons

Pros: More concise than XML, JSON parsing is faster, making it a preferred choice for web APIs.

Cons: Lacks some features such as comments and is less suitable for very complex data structures compared to XML.

JSON Data vs Tabular Representation

```
[
  {
    "title": "Big Data for Dummies",
    "author": "John Doe",
    "year": 2021,
    "price": 29.99
  },
  {
    "title": "Data Science 101",
    "author": "Jane Smith",
    "year": 2020,
    "price": 35.95
  }
]
```

Title	Author	Year	Price
Big Data for Dummies	John Doe	2021	29.99
Data Science 101	Jane Smith	2020	35.95

Other Data Transfer Formats

- **CSV (Comma-Separated Values):** Simple format for tabular data. Not suitable for complex hierarchical data.
- **YAML (YAML Ain't Markup Language)**: Human-friendly data serialization standard, good for configuration files.
- **Protocol Buffers (Protobuf)**: Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data, similar to XML but smaller, faster, and simpler.
- **Avro**: Developed within the Apache Hadoop project, supports schema evolution.
- **Parquet**: Columnar storage format, optimized for work with complex nested data structures.

Considerations for Choosing a Data Transfer Format

- **Human Readability**: Important for debugging and transparency.
- **Data Complexity**: Hierarchical formats like XML may be necessary for complex data structures.
- **Performance**: Binary formats like Protobuf are typically faster to parse and smaller in size.
- **Ecosystem Integration**: The choice might depend on the systems and environments in use.
- **Schema Evolution**: Systems that require changes over time benefit from formats like Avro that handle schema evolution well.