Data Analytics with FHIR: Scalability and Security

Gidon Gershinsky, IBM Research – Haifa Lab
Speaker

• Senior Architect at IBM Research
gidon@il.ibm.com

• Leading role in Apache Parquet community work on format and mechanism for secure data storage
  • folks from many companies are involved: IBM, Uber, Netflix, Cloudera, Emotiv, Vertica, Ursa Labs, Apple
    • October 2019: open standard for big data protection, parquet-format-2.7.0

• Number of projects on secure analytics with encrypted data
  • EU Horizon 2020: healthcare and connected car usecases
  • Apache Spark with Parquet encryption
    • working with Apache Spark community
Big Data Analytics

- Industry trend - instead of DB, keep big data in files (columnar format), and use analytic engines
  - Unlimited scalability, cheap storage and top speed of data processing
  - Beyond SQL: Machine learning / AI tools
- Technologies
  - Analytic Engine: Apache Spark
  - Big Data storage: Apache Parquet
  - Big Data protection (encryption and integrity verification): Apache Parquet Modular Encryption
- “RestAssured” –
  EU Horizon 2020 research project (N 731678)
  - Project partners
    IBM, Adaptant, OCC, Thales, UDE, IT Innovation
  - Project usecases
    usage-based car insurance, social services

Usage-based Insurance Service

Spark

Parquet

Personal Data of Drivers
- driver ID (license #)
- name
- home address
- email address
- credit card

Car Monitoring Events
- car ID (plate #)
- driver ID (license #)
- speed
- location
Big Data Analytics in Healthcare

• “ProTego” – EU Horizon 2020 research project (N 826284)
  • Project partners: St Raffaele hospital, Marina Salud hospital, IBM, GFI, ITI, UAH, IMEC, KUL, ICE
  • Analytics on sensitive healthcare data
  • HL7 FHIR and Parquet standards

• Analytic Engine: Apache Spark
• Big FHIR Data storage: Apache Parquet
• Big FHIR Data protection: Apache Parquet Modular Encryption
  • privacy: hide personal data
  • integrity: prevent tampering with patient health data
Why Apache Spark

• “Apache Spark™ is a unified analytics engine for large-scale data processing”

• Speed
  • “Run workloads 100x faster”
  • “.. state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine” – and heavy use of RAM

• Ease of Use
  • “Write applications quickly in Java, Scala, Python, R, and SQL”

• Generality
  • “Combine SQL, streaming, and complex analytics”
What is Apache Parquet

• Default file format in Apache Spark for analytic data

• Leveraged by every major tech company (such as Apple, Uber, IBM, Amazon, Microsoft)

• Besides Apache Spark, integrated in basically every analytic framework and query engine (Hive, Presto, pandas, Redshift Spectrum, Impala, etc)
Why Apache Parquet

• Columnar format, with column filtering (projection) - fetch only the columns you need from storage

• Predicate push down – filter rows by column (chunk or page) minimum and maximum – for each relevant column, fetch only pieces you need from storage -or skip fetching full file(s)

• Compression and encoding – store and fetch data efficiently

• Nested column support – map JSON objects (FHIR!)
Parquet Modular Encryption: Goals

• Protect sensitive data-at-rest (in storage)
  • data privacy/confidentiality: encryption - hiding sensitive information
  • data integrity: tamper-proofing sensitive information
  • in any storage - untrusted, cloud or private, file system, object store, archives

• Preserve performance of analytic engines
  • full Parquet capabilities (columnar projection, predicate pushdown, etc) with encrypted data

• Leverage encryption for fine-grained access control
  • per-column encryption keys
  • key-based access in any storage: private -> cloud -> archive
Parquet Encryption: Privacy

- Hiding sensitive information (from unauthorized parties)

- Full encryption: all data and metadata modules
  - min/max values, schema, encryption key ids, list of sensitive columns

- Separate keys for sensitive columns
  - column data and metadata
  - column access control

- Storage server / admin never sees encryption keys or unencrypted data
  - “client-side” encryption
Parquet Encryption: Integrity Guarantees

- **File data and metadata are not tampered with**
  - modifying data page contents
  - replacing one data page with another

- **File not replaced with wrong file**
  - unmodified - but e.g. outdated
  - sign file contents and file id

- Example: altering **healthcare** data - patient record or medical sensor readings

- **AES GCM**: “authenticated encryption”
  - implemented in CPU hardware
Performance Effect of Encryption

- AES ciphers implemented in CPU hardware (AES-NI)
  - Gigabyte(s) per second
  - Order(s) of magnitude faster than “application stack” (App/Framework/Parquet/compression/IO)
- C++: OpenSSL EVP libraries tap into AES-NI directly
- Java: AES-NI support in HotSpot since Java 9
  - Java 11.0.4 – enhanced AES GCM decryption
- Parquet min units (pages) are encrypted, not individual values
  - 0.003% size overhead
  - page size: ~1MB => maximal encryption speed
    - orders of magnitude faster than encrypting each value
- Sensitive columns: ~ one in ten in a typical table
  - further reduction in encryption time

Benchmark example

- Java 11.0.4
- Intel Core i7
- Parquet with SNAPPY compression
- AES_GCM algorithm
- Decryption overhead
  - all (19) columns encrypted: 3.6%
  - 2 columns encrypted: 0.7%
  - Reader app that does nothing (blackhole)
  - real apps: lower overhead!
Spark with Parquet Encryption

• No changes in Spark code
  • For example: Spark 2.3.0 - replace Parquet-1.8.2 with Parquet-1.8.2-E (a couple of jar files)

• Writing Parquet files in standard encryption format
  • parquet-format-2.7.0, released in Oct’19

• Invoke encryption via Hadoop parameters
  • Hadoop configuration already passed from Spark to Parquet

• KMS and envelope encryption supported
Hadoop Encryption Parameters

- * prototype – subject to change

**Mandatory parameters**

- "encryption.column.keys"
  - list of columns to encrypt, with master key IDs.

- "encryption.footer.key"
  - master key ID for footer encryption/signing

- "encryption.kms.client.class"
  - name of class implementing KmsClient interface

- "encryption.key.access.token"
  - auth token that will be passed to KMS

**Optional parameters**

- "encryption.algorithm"
- "encryption.file.id"
  - file replacement protection
- "encryption.plaintext.footer"

- "<masterKeyID>:<colName>,<colName>;<masterKeyID>:<colName>, .."
  - jointly defined for Parquet and ORC column encryption

HIVE-21848
Hadoop Decryption Parameters

- Less parameters than for encryption

**Mandatory parameters**

- "encryption.kms.client.class"
  - name of class implementing KmsClient interface
- "encryption.key.access.token"
  - auth token that will be passed to KMS

**Optional parameters**

- "encryption.file.id"
  - file replacement protection
Today: Analytics with FHIR Database backend

Data Producer

FHIR put

Sensors

FHIR Server

FHIR bulk export

Database

(ND) JSON

Parquet

第二出口

Spark

SQL

SQL (or ML)

Medical care givers, Researchers
FHIR Analytics with Parquet backend

Data Producer

Sensors

FHIR put

FHIR Server

Database

Regular data (patients, etc)

Big data (sensor observations, etc)

Parquet

Medical care givers, Researchers

SQL (or ML)

Apache Spark™
FHIR Analytics with Parquet backend and Bulk Export

Data Producer → FHIR put → FHIR Server → Parquet

FHIR bulk export: Parquet

Database: Regular data (patients, etc)

Database: Big data (sensor observations, etc)

SQL (or ML)

Medical care givers, Researchers
Parquet as one of FHIR Bulk Formats

- [https://hl7.org/fhir/formats.html](https://hl7.org/fhir/formats.html)
  - Bulk Data Formats
    - “Apache Parquet/Avro (bulk data formats under consideration)”

- Why Parquet (with any storage backend, including DateBases)
  - much less bytes to transfer: encoding, compression
  - built-in security: encryption and tamper-proofing. no need in TLS.
  - ~10x-100x faster analytics: column projection and predicate pushdown
    - if encrypted – no overhead! no need in decrypting full files.
  - JSON (FHIR) mapping to Parquet nested columns

- Why Parquet (with Parquet backend)
  - zero cost export, just send the files (* for direct dump)
    - if encrypted – update file metadata (no need to re-encrypt data)
Feedback and Questions!