Structural Bioinformatics Training Workshop & Hackathon 2017

Advanced MMTF-Spark

Peter Rose

Structural Bioinformatics Laboratory
San Diego Supercomputer Center
UC San Diego

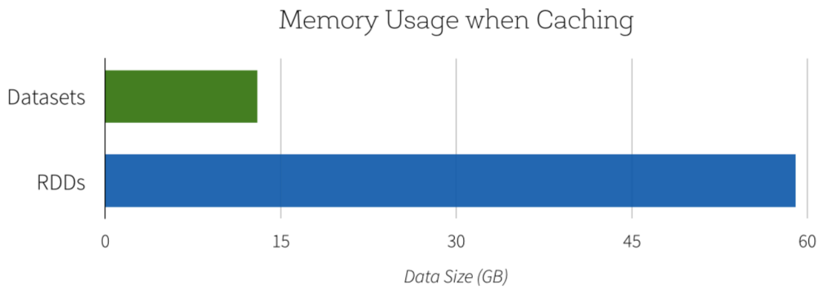
Introduction

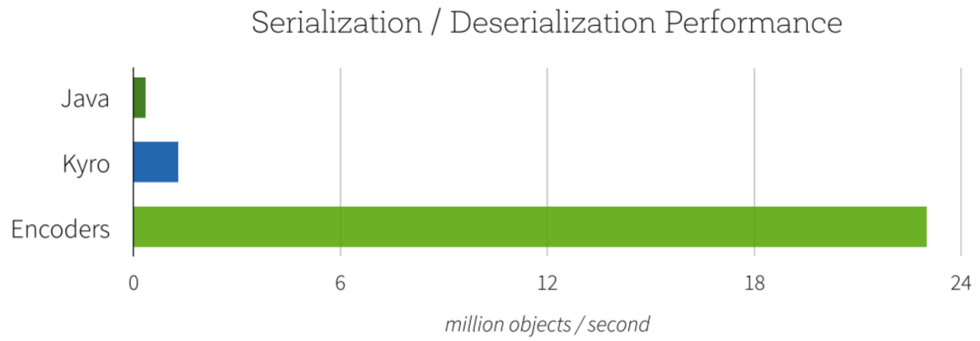
- **Spark SQL and Dataset API**
  - Augmenting MMTF data with annotations from PDB and other 3rd party resources
  - Creating datasets of molecular interactions
  - Querying and analyzing datasets

- **MMTF Data Structure**
  - Introduction to StructureDataInterface
  - Traversing the structural hierarchy

Spark Dataset

- Table of typed objects with a relational schema
- Similar to Python Pandas and R Dataframes
- Distributed data structure optimized for performance
- Distributed SQL queries on Dataset (Spark SQL)





Source: https://databricks.com/blog/2016/01/04/introducing-apache-spark-datasets.html

Custom Report of PDB Annotations

```
// spark setup
JavaSparkContext sc = …

// retrieve PDB annotation: Binding affinities (Ki, Kd),
// group name of the ligand (hetId), and the
// Enzyme Classification number (ecNo)
Dataset<Row> ds = CustomReportService.getDataset("Ki","Kd","hetId","ecNo");

// show the schema of this dataset
ds.printSchema();

// select structures that either have a Ki or Kd value(s) and
// are protein-serine/threonine kinases (EC 2.7.1.*)
// by using dataset operations
ds = ds.filter("(Ki IS NOT NULL OR Kd IS NOT NULL) AND ecNo LIKE '2.7.11.%'");
ds.show(10);
```

Capacity limitation: do not request more than 4 fields per dataset

List of custom report fields: http://www.rcsb.org/pdb/results/reportField.do

Creating a Temporary Table/SQL

```
// spark setup
JavaSparkContext sc = …

// retrieve PDB annotation: Binding affinities (Ki, Kd),
// group name of the ligand (hetId), and the
// Enzyme Classification number (ecNo)
Dataset<Row> ds = CustomReportService.getDataset("Ki","Kd","hetId","ecNo");

// select structures that either have a Ki or Kd value(s) and
// are protein-serine/threonine kinases (EC 2.7.1.*)
// by creating a temporary query and running SQL
ds.createOrReplaceTempView("table");
ds.sparkSession().sql("SELECT * from table WHERE
(Ki IS NOT NULL OR Kd IS NOT NULL) AND ecNo LIKE '2.7.11.%'");

ds.show(10);
```

List of custom report fields: http://www.rcsb.org/pdb/results/reportField.do

Problem 1

- **Retrieve and query a Dataset**
  - Navigate to project: 4-advanced-spark in Eclipse
  - Find and and open Problem01.java (src/main/java)
  - Look at // TODO for the problem description
  - Insert your code after the // TODO and run it

Problem 2

- **Join two datasets**
  - Navigate to project: 4-advanced-spark in Eclipse
  - Find and and open Problem02.java (src/main/java)
  - Look at // TODO for the problem description
  - Insert your code after the // TODO and run it

Problem 3

- **Create and query a new dataset**
  - Navigate to project: 4-advanced-spark in Eclipse
  - Complete the code in UnitCellExtractorProblem03.java
  - Complete the code in Problem03.java
  - Then run Problem03.java

Find Interactions

```
// use a representative subset of the PDB (1st member of each sequence cluster)
int sequenceIdentity = 40;
pdb = pdb.filter(new BlastClusters(sequenceIdentity));

double cutoffDistance = 3.0;
GroupInteractionExtractor finder =
                         new GroupInteractionExtractor("ZN", cutoffDistance);

Dataset<Row> interactions = finder.getDataset(pdb).cache();
interactions.printSchema();

System.out.println("# interactions: " + interactions.count());

// list some example interactions
interactions.show(20);
```

Information about BlastClust: ftp://resources.rcsb.org/sequence/clusters/

Analyze Interactions

```
// note, this static import is required for this example
import static org.apache.spark.sql.functions.col;

// use a representative subset of the PDB (1st member of each sequence cluster)
int sequenceIdentity = 40;
pdb = pdb.filter(new BlastClusters(sequenceIdentity));

double cutoffDistance = 3.0;
GroupInteractionExtractor finder =
                         new GroupInteractionExtractor("ZN", cutoffDistance);

Dataset<Row> interactions = finder.getDataset(pdb).cache();

// show the top 10 interacting groups
interactions
        .groupBy(col("residue2"))
        .count()                         // count by residue type
        .sort(col("count").desc())  // sort descending
        .show(10);
```

Analyze Interactions Continued

```
long n = interactions.count();
System.out.println("Top interacting group/atoms types");

Dataset<Row> topGroupsAndAtoms = interactions
                .filter("element2 != 'C'") // exclude carbon interactions
                .groupBy("residue2","atom2")
                .count();

topGroupsAndAtoms
        .withColumn("frequency", col("count").divide(n)) // add frequency col.
        .filter("frequency > 0.01") // filter out occurrences < 1 %
        .sort(col("frequency").desc()) // sort descending
        .show(20);
```

Demo 1

- **Show results of interaction analysis**
  - https://github.com/sbl-sdsc/mmtf-spark/blob/master/src/main/java/edu/sdsc/mmtf/spark/datasets/demos/InteractionAnalysisSimple.java
  - https://github.com/sbl-sdsc/mmtf-spark/blob/master/src/main/java/edu/sdsc/mmtf/spark/datasets/demos/InteractionAnalysisAdvanced.java

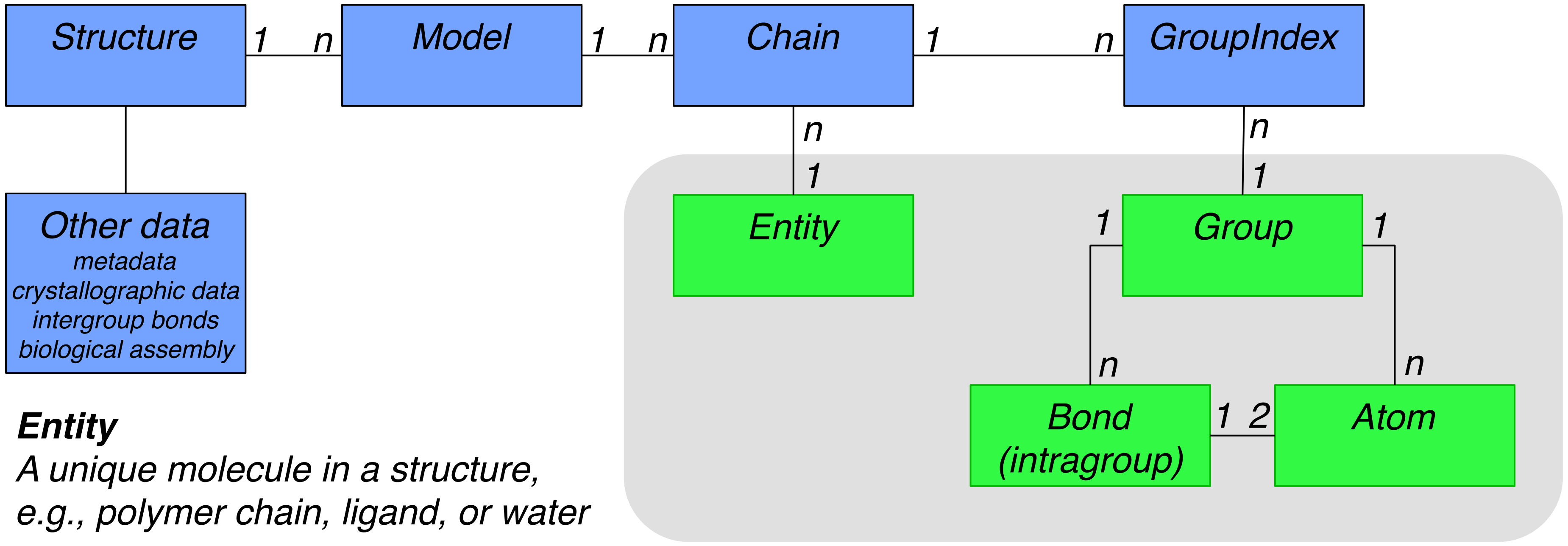
Problem 4

- **Analyze the interactions of the terminal phosphate in ATP with protein-serine/threonine kinases**
  - Navigate to project: 4-advanced-spark in Eclipse
  - Complete and run the code in Problem04.java

MMTF API: StructureDataInterface

Uncompressed MMTF data are accessible through the StructureDataInterface

StructureDataInterface is a flat (columnar encoded) data structure with an implicit hierarchy



Entity
A unique molecule in a structure, e.g., polymer chain, ligand, or water

Group
A unique chemical group (residue)

unique entities and groups are stored only once e.g., 20 natural amino acids, water

Demo 2

- **How to traverse the structural hierarchy**
  - https://github.com/sbl-sdsc/mmtf-spark/blob/master/src/main/java/edu/sdsc/mmtf/spark/analysis/TraverseStructureHierarchy.java

Problem 5

- **Traverse the structural hierarchy and calculate the molecular weight of a structure**
  - Navigate to project: 4-advanced-spark in Eclipse
  - Complete and run the code in Problem05.java

Summary

- **Spark Dataset API provides an efficient distributed tabular data structure**
- **Can be queried using Spark SQL**
- **We used datasets to**
  - get additional annotations not available in MMTF
  - store and query the results of structural calculations
- **We learned how to traverse the MMTF StructureDataInterface**

Resources

- **Spark SQL, DataFrames and Datasets Guide**
  - https://spark.apache.org/docs/latest/sql-programming-guide.html
- **MMTF Website**
  - http://mmtf.rcsb.org
- **GitHub Repository**
  - https://github.com/sbl-sdsc/mmtf-spark
- **MMTF File Format**
  - Bradley AR, et al. (2017) MMTF—An efficient file format for the transmission, visualization, and analysis of macromolecular structures. PLOS Computational Biology 13(6): e1005575. https://doi.org/10.1371/journal.pcbi.1005575
  - Valasatava Y, et al. (2017) Towards an efficient compression of 3D coordinates of macromolecular structures. PLOS ONE 12(3): e0174846. https://doi.org/10.1371/journal.pone.0174846
- **RCSB PDB Web Services and Query System**
  - Rose, PW, et al. (2013) The RCSB Protein Data Bank: new resources for research and education, Nucleic Acids Res 41: D475-D482. https://doi.org/10.1093/nar/gks1200
  - Rose, PW, et al. (2011) The RCSB Protein Data Bank: redesigned web site and web services, Nucleic Acids Res 39: D392-D401. https://doi.org/10.1093/nar/gkq1021