INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Knowledge Engineering and Data Mining

Dr. Animesh Chaturvedi

Assistant Professor: **IIIT Dharwad**

Young Researcher: **Heidelberg Laureate Forum**

and **Pingala Interaction in Computing**

Young Scientist: **Lindau Nobel Laureate Meetings**

Postdoc: **King's College London** & **The Alan Turing Institute**

PhD: **IIT Indore** MTech: **IIITDM Jabalpur**

LINDAU NOBEL LAUREATE MEETINGS

HEIDELBERG LAUREATE FORUM

KING'S College LONDON

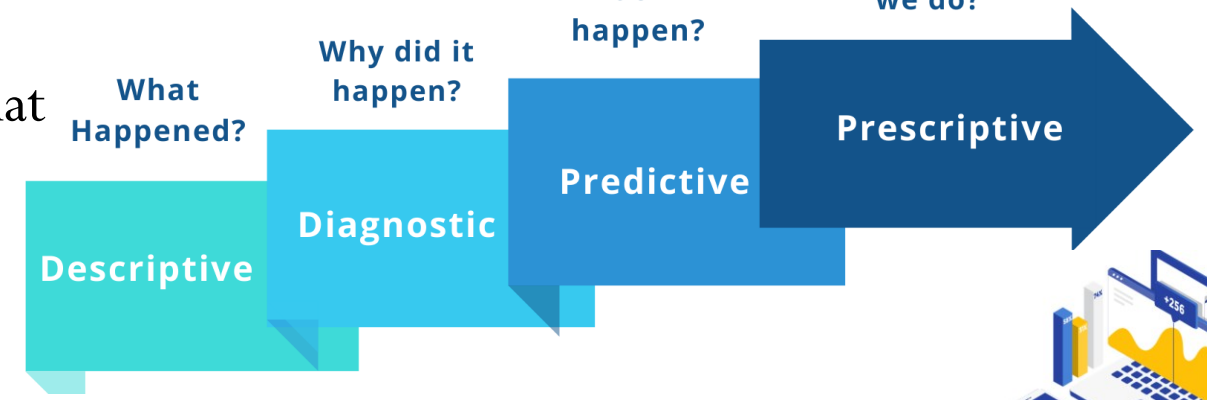
The Alan Turing Institute

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

Data Science and Data Analytics

- **Data Science:** interdisciplinary science that
  - deals with data: methods, algorithms, processes, systems etc.
  - Theory oriented
- **Data Analytics:** analysis of data that
  - discovers trends, graph, tables etc.
  - Technology oriented
- Both
  - extracts knowledge and apply actions through structured and unstructured data insights
  - deals with data mining, machine learning, data management, and big data.
- **Data Scientist** and **Data Analyst** applies Data Science and Data Analytics



Evolution to Data Science Societies

- Institute of Radio Engineers (**IRE**) in 1951
- Institute of Electrical and Electronics Engineers (**IEEE**), **IEEE Computer Society** (1946; 1963)
- Association for Computing Machinery (**ACM**) in 1947
- Technology Engineering and Management Society (**TEMS**) 1955
- IEEE Systems, Man, and Cybernetics Society (**SMC**) (1958; 1972)
- Association for Computational Linguistics (**ACL**) 1962
- International Joint Conf. on Artificial Intelligence (**IJCAI**) 1969
- Special Interest Group on Management of Data (**SIG-MOD**), 1975
- Special Interest Group on Information Retrieval (**SIG-IR**) 1978
- Association for the Advancement of Artificial Intelligence (**AAAI**) 1979
- International Conf. on Machine Learning (**ICML**) 1980 in Pittsburgh
- Conf. on Neural Information Processing Systems (**NeurIPS**) 1986 -1987
- IEEE International Conf. on Data Engineering (**IEEE ICDE**) 1984
- International World Wide Web Conference (**WWW**) - Web Conference 1994
- International Conf. on Knowledge Discovery and Data Mining (**SIG-KDD**) 1995
- IEEE International Conf. on Data Mining (**IEEE ICDM**) 2001
- IEEE International Conf. on Big Data (**IEEE Big Data**) 2013
- IEEE International Conf. on Data Science and Advanced Analytics (**IEEE DSAA**) 2014

IRE
IEEE
ACM
TEMS
SMC
ACL
IJCAI
SIG-MOD & IR
AAAI, ICML
NeurIPS
ICDE, WWW
KDD, ICDM
Big Data,
DSAA

DBLP & Google Scholar

- "Data" in DBLP search*
  - There are 459+ Venues matched in the DBLP, world most referred computer science bibliography website.
  - DBLP launched in 1993

- "Data" in Google Scholar metrics
  - Top 20 publications venues matching "data"

	Key Publications
1.	ACM SIGKDD International Conference on **Knowledge Discovery & Data Mining**
2.	IEEE Transactions on **Knowledge and Data Engineering**
3.	International Conference on **Artificial Intelligence and Statistics**
4.	ACM International Conference on **Web Search and Data Mining**
5.	Journal of **Big Data**
6.	IEEE International Conference on **Data Mining**
7.	IEEE International Conference on **Big Data**
8.	**Knowledge and Information Systems**
9.	ACM Conference on **Recommender Systems**
10.	Wiley Interdisciplinary Reviews: **Data Mining and Knowledge Discovery**
11.	European Conference on **Machine Learning and Knowledge Discovery in Databases**
12.	ACM Transactions on **Intelligent Systems and Technology** (TIST)
13.	**Data Mining and Knowledge Discovery**
14.	SIAM International Conference on **Data Mining** (SDM)
15.	ACM Transactions on **Knowledge Discovery from Data** (TKDD)
16.	International Conference on Advances in **Social Networks Analysis and Mining**
17.	**Big Data Mining and Analytics**
18.	International Journal of **Data Science and Analytics**
19.	**Social Network Analysis and Mining**
20.	Pacific-Asia Conference on **Knowledge Discovery and Data Mining** (PAKDD)
21.	IEEE International Conf. on Data Science and Advanced Analytics (**IEEE DSAA**)

Knowledge Discovery and Data Mining

- "a unifying framework for Knowledge Discovery in Database (KDD)"
- links between data mining, knowledge discovery, and other related field

Selection, Preprocessing, Transformation, Data Mining, Interpretation/Evaluation

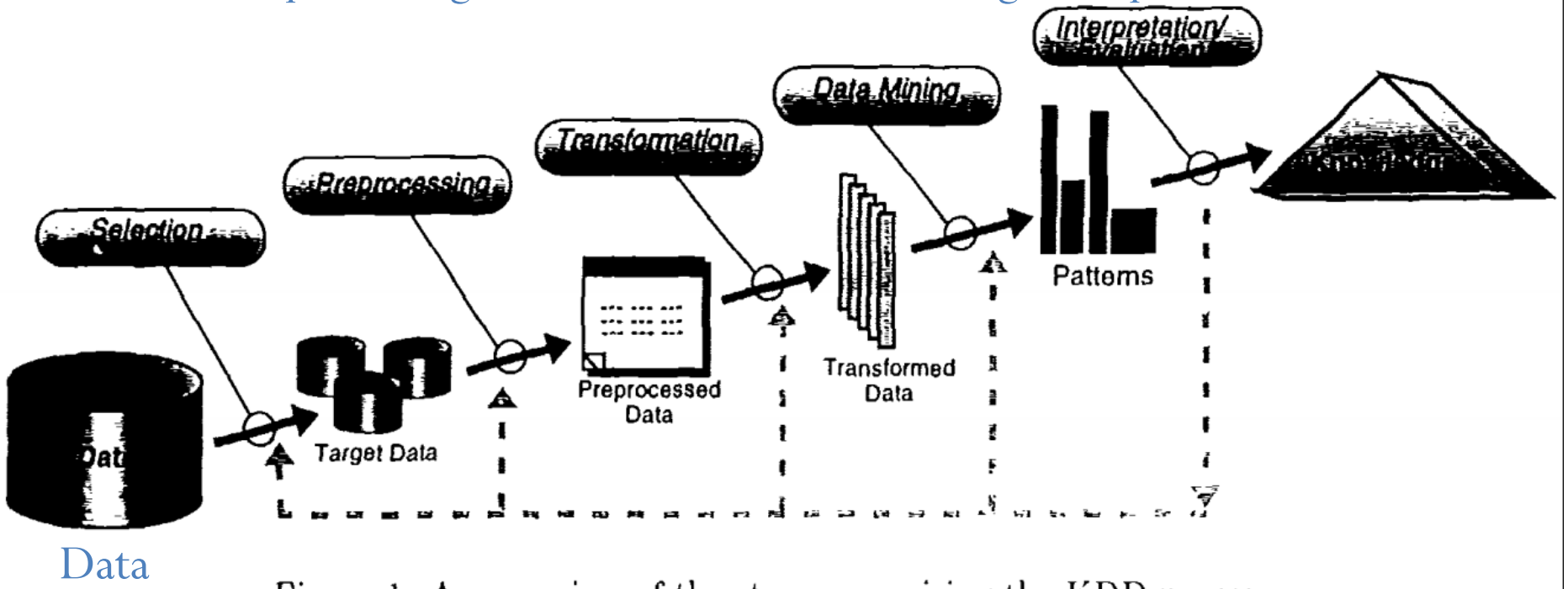


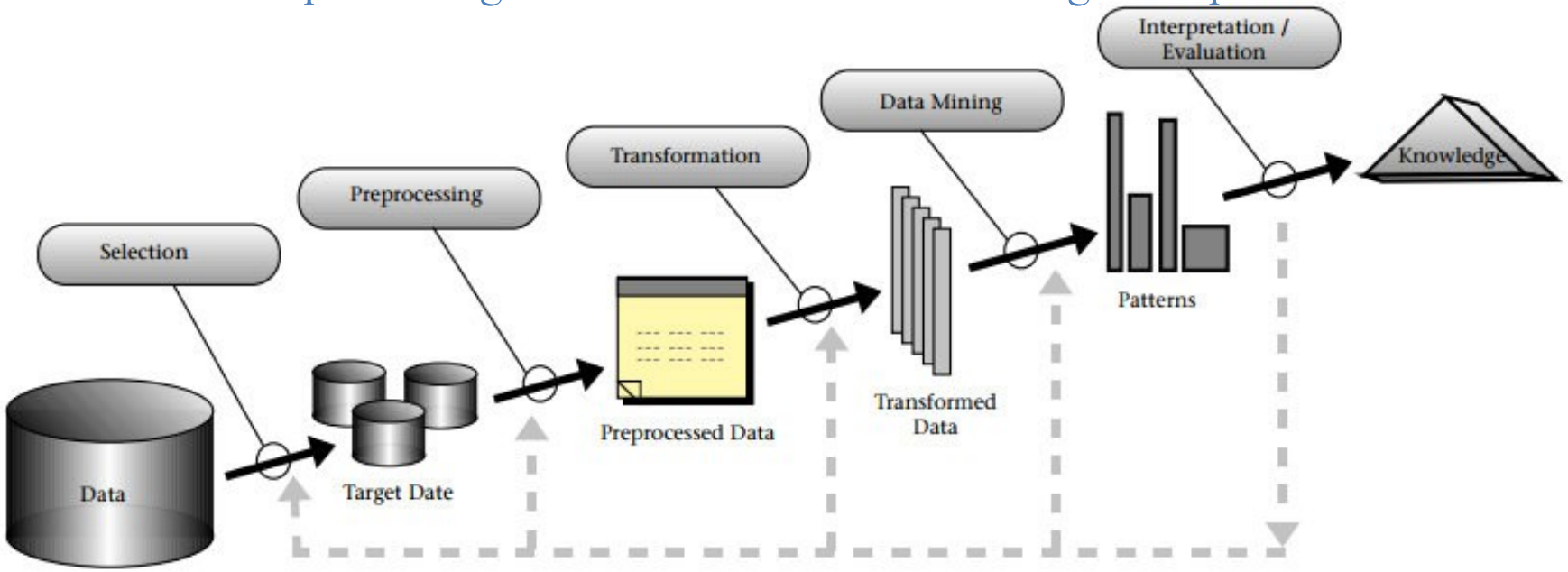
Figure 1: An overview of the steps comprising the KDD process.

Fayyad, Usama M., Gregory Piatetsky-Shapiro, and Padhraic Smyth.
"Knowledge Discovery and Data Mining: Towards a Unifying Framework." ***KDD*. 1996.**

Knowledge Discovery and Data Mining

- "a unifying framework for Knowledge Discovery in Database (KDD)"
- links between data mining, knowledge discovery, and other related field

Selection, Preprocessing, Transformation, Data Mining, Interpretation/Evaluation



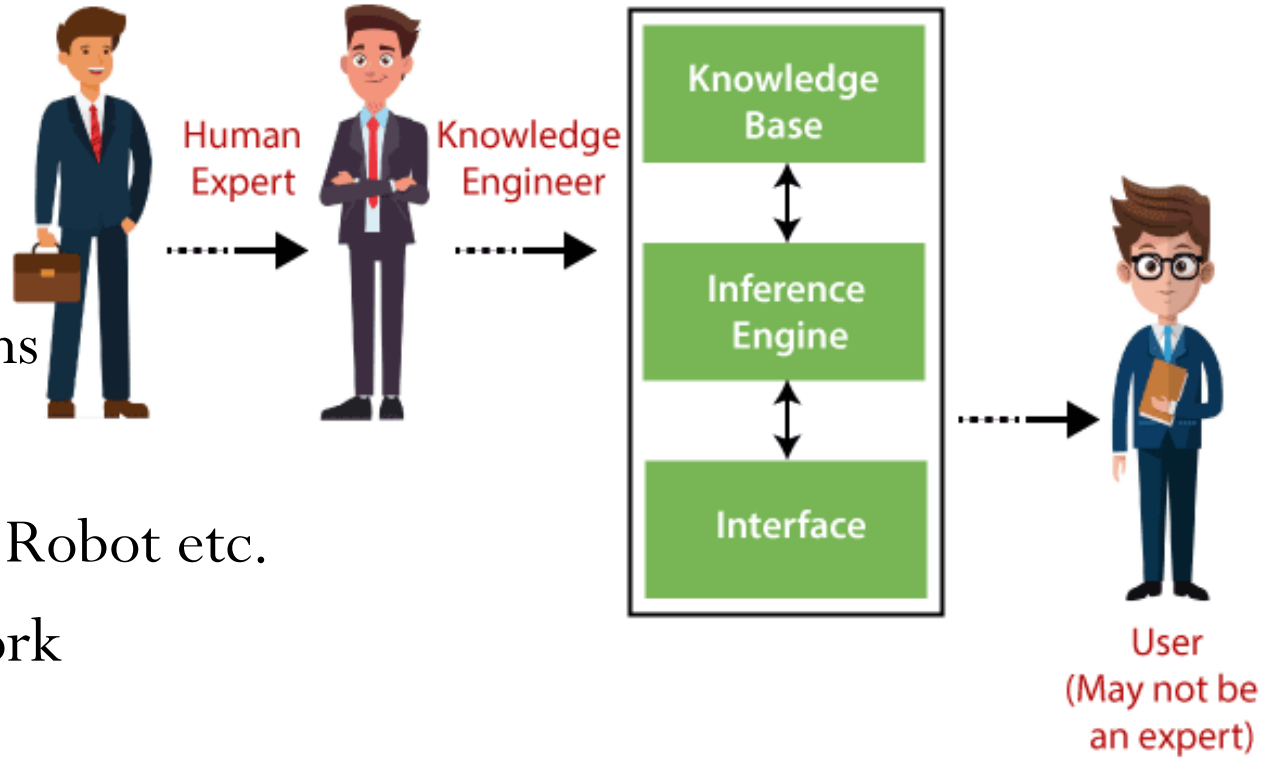
Data

https://www.kdnuggets.com/2016/03/data-science-process-rediscovered.html/2

Fayyad, Usama M., Gregory Piatetsky-Shapiro, and Padhraic Smyth.
"Knowledge Discovery and Data Mining: Towards a Unifying Framework." ***KDD*. 1996.**

Expert, System, Data, and Knowledge Engineer

- Expert Systems (1960-74)
  - Explicit rules
- Playing Chess,
- Organic and Biology recommendations
- Solving word problem in Algebra
- Natural Language processing, Mobile Robot etc.
- Backpropagation based Neural Network
- **Association Rule Mining**
  by Rakesh Agrawal and Srikant Ramakrishnan 1993-95
- **Big Files**, **Google File System**, **Map-Reduce** in 1995-2005
  by Larry Page, Sergey Brin, Sanjay Ghemawat, Jeff Dean et al.



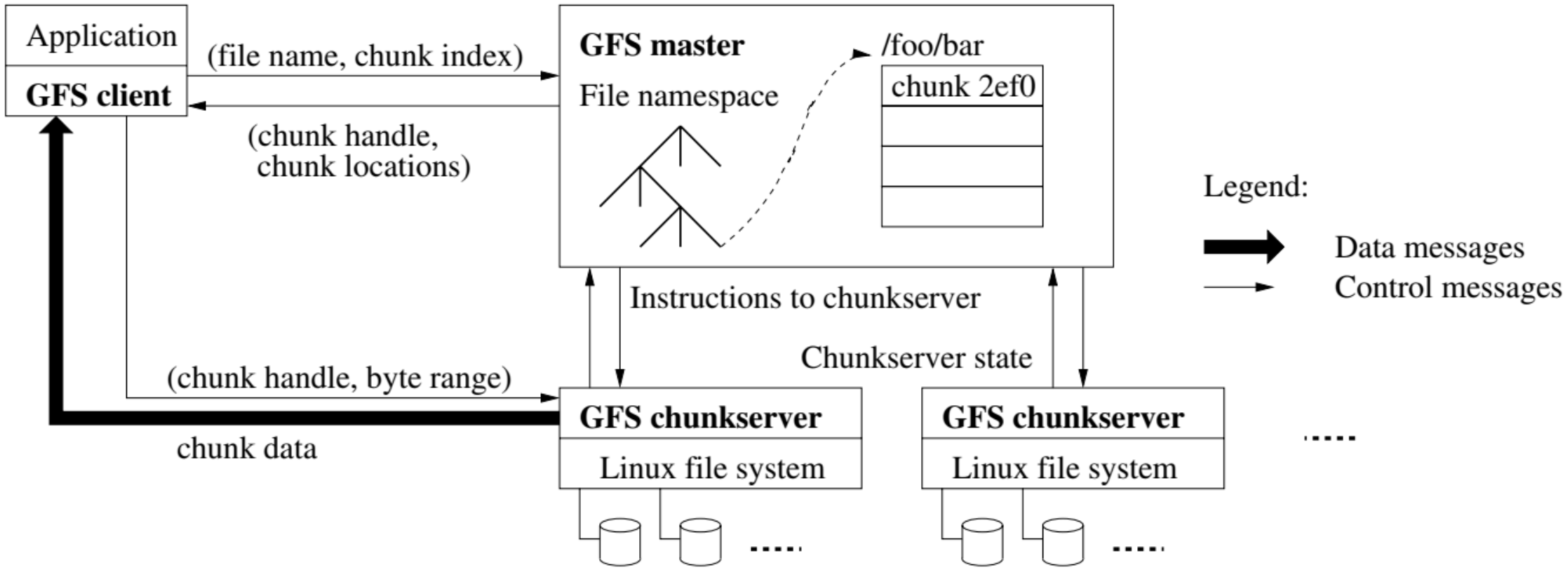
Big Files to Google File System (GFS)

- Earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin.
  - Supervisors: Hector Garcia-Molina, Rajeev Motwani, Jeff Ullman, and Terry Winograd
- “Big File” was regenerated as “Google File System” by Sanjay Ghemawat, et al.
- Google File System (GFS)
  - “It is widely deployed within Google as the storage platform for the generation and processing of data used by Google service as well as research and development efforts that require large data sets.” 2003
  - “The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.” 2003

http://infolab.stanford.edu/~backrub/google.html

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

Google File System (GFS)



https://en.wikipedia.org/wiki/Google_File_System

https://sites.google.com/site/gfsassignmentwiki/home

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

GFS to HDFS

- Google File System (GFS) has similar open-source
  - "Hadoop Distributed File System (HDFS)"
- GFS and HDFS are distributed computing environment to process "Big Data".
- GFS and HDFS are not implemented in the kernel of an operating system, but they are instead provided as a userspace library.
- GFS and HDFS properties
  - Files are divided into fixed-size chunks of 64 megabytes.
  - Scalable distributed file system for large distributed data intensive applications.
  - Provides fault tolerance.
  - High aggregate performance to a large number of clients.

https://en.wikipedia.org/wiki/Google_File_System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

Big Data

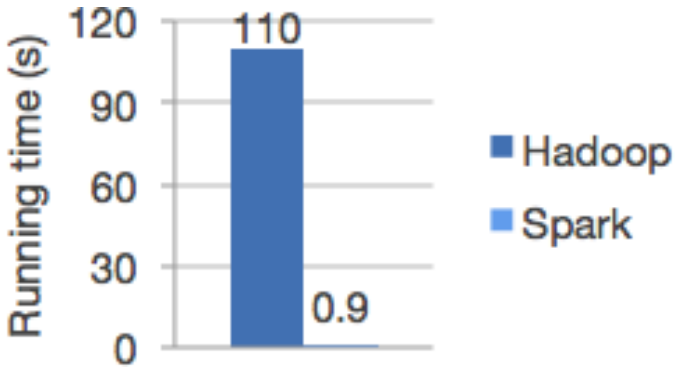
- Big data can be described by the following characteristics:
  - Volume: size large than terabytes and petabytes
  - Variety: type and nature, structured, semi-structured or unstructured
  - Velocity: speed of generation and processing to meet the demands
  - Veracity: the data quality and the data value
  - Value: Useful or not useful
- The main components and ecosystem of Big Data
  - Data Analytics: data mining, machine learning and natural language processing etc.
  - Technologies: Business Intelligence, Cloud computing & Databases etc.
  - Visualization: Charts, Graphs etc.

Hadoop to handle Big Data

- "The Apache Hadoop project develops open source software for reliable, scalable, distributed computing." Software library and a framework.
- Created by Doug Cutting Named on his son's stuffed elephant
- For distributed processing of large data sets across clusters of computers using simple programming models.
- Locality of reference
- Scalability: Scale up from single servers to thousands of machines,
  - Each offering local computation and storage
  - Program remains same for 10, 100, 1000,… nodes
  - Corresponding performance improvement
- Fault-tolerant file system:
  - Detect and handle failures and Delivering a highly-available.
- Hadoop Distributed File System (HDFS) Modeled on Google File system
- MapReduce for Parallel computation using
- Components – Pig, Hbase, HIVE, ZooKeeper

Apache Spark

- Unified analytics engine for large-scale data processing.
- Speed: Run workloads 100x faster.
- Both batch and streaming data, using Directed Acyclic Graph (DAG) scheduler, a query optimizer, and a physical execution engine.
- Ease of Use: Write applications quickly in Java, Scala, Python, R, and SQL.
- Spark offers 80+ high-level operators to build parallel apps.



https://spark.apache.org/

Spark: Unified Big Data Analytics

- New applications of Big data workloads on Unified Engine of
  - Streaming, Batch, and Interactive.
- Composability in programming libraries for big data and encourages development of interoperable libraries
- Combining the SQL, machine learning, and streaming libraries in Spark

```
// Load historical data as an RDD using Spark SQL
val trainingData = sql(
   "SELECT location, language FROM old_tweets")

// Train a K-means model using MLlib
val model = new KMeans()
   .setFeaturesCol("location")
   .setPredictionCol("language")
   .fit(trainingData)
// Apply the model to new tweets in a stream
TwitterUtils.createStream(...)
      .map(tweet => model.predict(tweet.location))
```

Zaharia, Matei, et al. "Apache spark: a unified engine for big data processing." *Communications of the ACM* 59.11 (2016): 56-65.

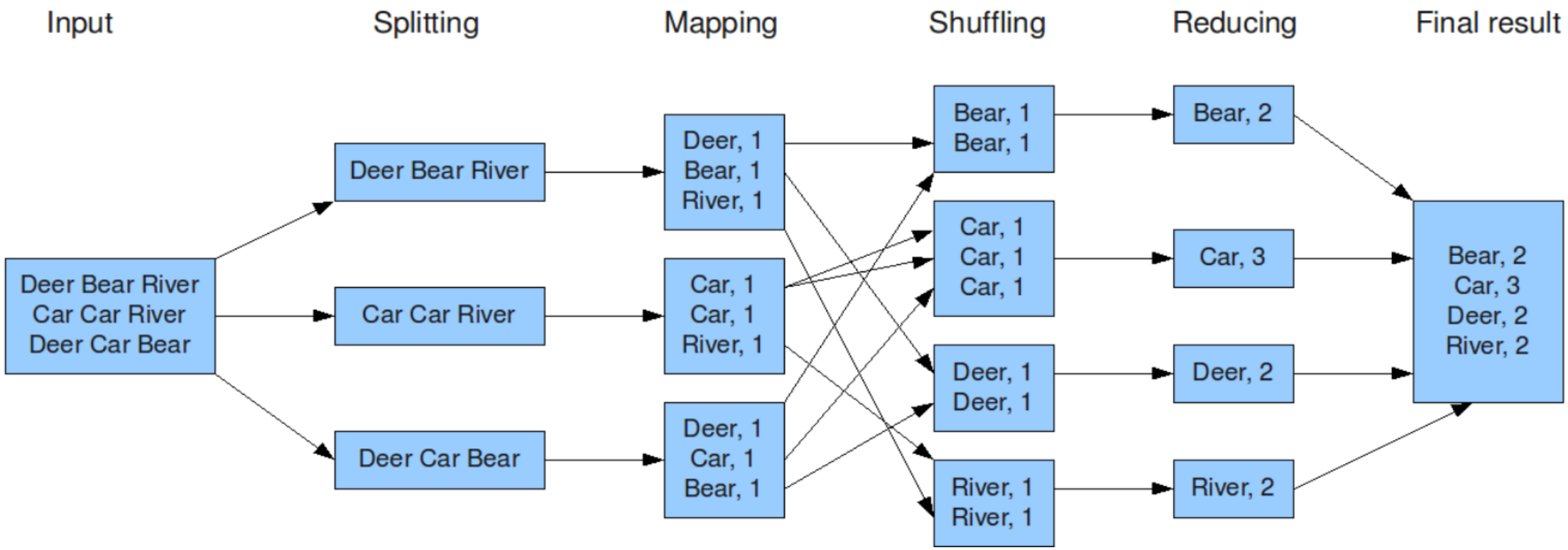
Apache Spark

- Combine SQL, streaming, and complex analytics. Spark libraries
  - SQL and DataFrames,
  - MLlib for machine learning,
  - GraphX, and
  - Spark Streaming.
- Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud.
- Run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes.
- Access data in HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources.

https://spark.apache.org/

Word Count

- **Map:** Input lines of text to breaks them into words gives outputs for each word <key = word, value =1 >
- **Reduce:** Input <word, 1> output <word, + value>



Spark Code Example

- Word Count

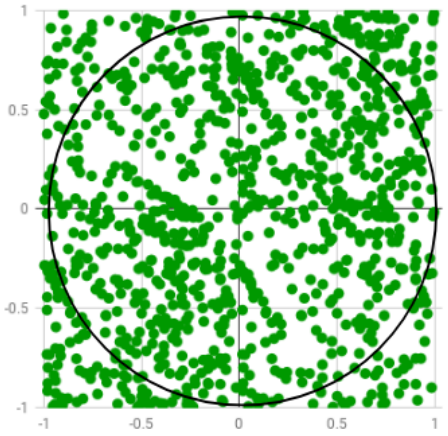
```
JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaPairRDD<String, Integer> counts = textFile
    .flatMap(s -> Arrays.asList(s.split(" ")).iterator())
    .mapToPair(word -> new Tuple2<>(word, 1))
    .reduceByKey((a, b) -> a + b);
counts.saveAsTextFile("hdfs://...");
```

- Pi Estimation

```
List<Integer> l = new ArrayList<>(NUM_SAMPLES);
for (int i = 0; i < NUM_SAMPLES; i++) {
  l.add(i);
}

long count = sc.parallelize(l).filter(i -> {
  double x = Math.random();
  double y = Math.random();
  return x*x + y*y < 1;
}).count();
System.out.println("Pi is roughly " + 4.0 * count / NUM_SAMPLES);
```

$$\text{Pi} = 4 \times \frac{number\ of\ random\ point\ inside\ the\ circle}{number\ of\ random\ point\ inside\ the\ square}$$



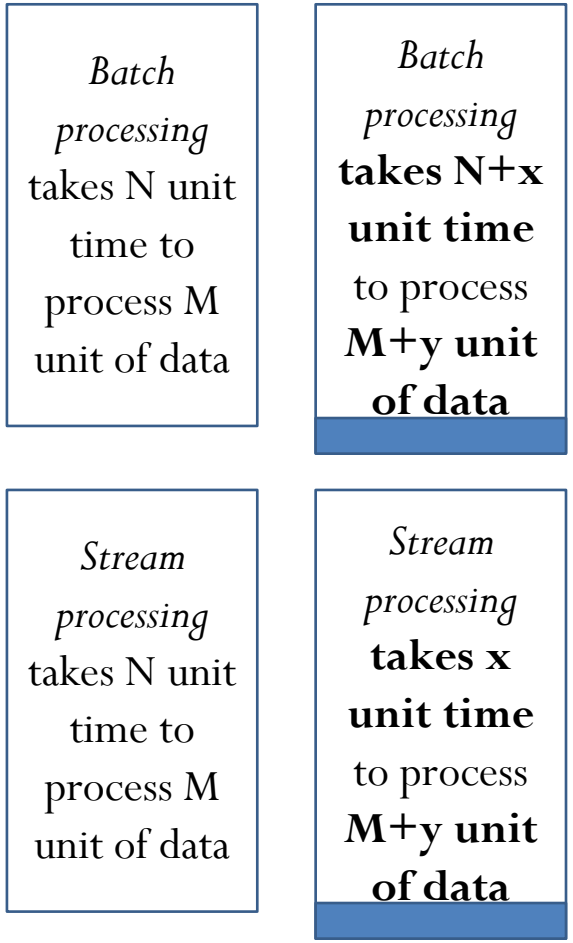
Spark Streaming

- Build scalable fault-tolerant streaming applications.
- Write streaming jobs -- Same Way -- Write batch jobs
  - Counting tweets on a sliding window

```
TwitterUtils.createStream(…)
.filter(_.getText.contains("Spark"))
.countByWindow(Seconds(5))
```

- Reuse the same code for batch processing
  - Find words with higher frequency than historic data:

```
stream.join(historicCounts).filter {
  case (word, (curCount, oldCount)) =>
    curCount > oldCount
}
```



https://spark.apache.org/streaming/

Spark MLlib

- Spark's scalable machine learning library
- Spark MLlib algorithms
  - Classification: logistic regression, naive Bayes,…
  - Regression: generalized linear regression, survival regression,…
  - Decision trees, Random forests, and Gradient-boosted trees
  - Recommendation: Alternating Least Squares (ALS)
  - Clustering: K-means, Gaussian mixtures (GMMs),…
  - Topic modeling: Latent Dirichlet Allocation (LDA)
  - **Frequent itemsets, Association rules, and Sequential pattern mining**

https://spark.apache.org/graphx/

Spark MLlib

- Spark's scalable machine learning library
- Spark MLlib algorithms
  - Classification: logistic regression, naive Bayes,…
  - Regression: generalized linear regression, survival regression,…
  - Decision trees, Random forests, and Gradient-boosted trees
  - Recommendation: Alternating Least Squares (ALS)
  - Clustering: K-means, Gaussian mixtures (GMMs),…
  - Topic modeling: Latent Dirichlet Allocation (LDA)
  - **Frequent itemsets, Association rules, and Sequential pattern mining**

https://spark.apache.org/graphx/

Frequent Patterns (FP)

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - itemset: A set of one or more items $I = \{i_1, i_2, \ldots . i_m\}$
  - Mining algorithms
    - Apriori: Association Rule Mining
    - FP-Growth
    - Parallel FP-Growth

Apriori: Association Rule Mining

- Let I = $\{i_1, i_2, \ldots . i_m\}$ be a set of literals, called items.
- *Support* of a rule X →Y is the percentage of transactions that contain both X and Y.
- *Confidence* of a rule is percentage the if-then statements (X →Y) are found true
- Find all rules that satisfy a user-specified *minimum Support* and *minimum Confidence*

TID	Transaction Items
1	Bread, Jelly, PeanutButter
2	Bread, PeanutButter
3	Bread, Milk, PeanutButter
4	Beer, Bread
5	Beer, Milk

[Bread] → [PeanutButter] (Sup = 60%, Conf = 75%)

[PeanutButter] → [Bread] (Sup = 60%, Conf = 100%)

[Beer] → [Bread] (Sup = 20%, Conf = 50%)

[PeanutButter] → [Jelly] (Sup = 20%, Conf = 33.33%)

[Jelly] → [PeanutButter] (Sup = 20%, Conf = 100%)

[Jelly] → [Milk] (Sup = 0%, Conf = 0%)

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIG-MOD. 1993.
Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules."VLDB 1995.

Support and Confidence

- Support count of X∪Y : Frequency or occurrence of an itemset X∪Y
  - Support is the fraction of transactions that contains X
- Confidence (association rule: X→Y)
  - Sup(X∪Y)/Sup(X) (conditional prob)
  - Confidence is conditional probability that a transaction having X also contains Y

TID	Transaction Items
1	Bread, Jelly, PeanutButter
2	Bread, PeanutButter
3	Bread, Milk, PeanutButter
4	Beer, Bread
5	Beer, Milk

[Bread] → [PeanutButter] (Sup = 60%, Conf = 75%)

[PeanutButter] → [Bread] (Sup = 60%, Conf = 100%)

[Beer] → [Bread] (Sup = 20%, Conf = 50%)

[PeanutButter] → [Jelly] (Sup = 20%, Conf = 33.33%)

[Jelly] → [PeanutButter] (Sup = 20%, Conf = 100%)

[Jelly] → [Milk] (Sup = 0%, Conf = 0%)

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIG-MOD. 1993.
Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules." VLDB 1995.

Support and Confidence

- Find all the rules X→Y with minimum Support and minimum Confidence
  - An itemset X∪Y is frequent if its support is no less than a minSup threshold

    $Sup(X \cup Y) \geq minSup$
  - A rule is significant if its confidence is no less than a minSup threshold

    $Sup(X \cup Y) / Sup(X) \geq minConf$

TID	Transaction Items
1	Bread, Jelly, PeanutButter
2	Bread, PeanutButter
3	Bread, Milk, PeanutButter
4	Beer, Bread
5	Beer, Milk

[Bread] → [PeanutButter] (Sup = 60%, Conf = 75%)

[PeanutButter] → [Bread] (Sup = 60%, Conf = 100%)

[Beer] → [Bread] (Sup = 20%, Conf = 50%)

[PeanutButter] → [Jelly] (Sup = 20%, Conf = 33.33%)

[Jelly] → [PeanutButter] (Sup = 20%, Conf = 100%)

[Jelly] → [Milk] (Sup = 0%, Conf = 0%)

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIG-MOD. 1993.
Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules." VLDB 1995.

Apriori: Association Rule Mining

- Find all rules that satisfy a user-specified *minimum Support* and *minimum Confidence*
  - 100% of transactions that purchase *PeanutButter* (antecedent) also purchase *Bread* (consequent).
  - Sup ≥ 60% (minimum Support) and Conf ≥ 75% (minimum Confidence) are the thresholds for rule generation
    - [Bread] ⟶ [PeanutButter] (Sup = 60%, Conf = 75%)
    - [PeanutButter] ⟶ [Bread] (Sup = 60%, Conf = 100%)

TID	Transaction Items
1	Bread, Jelly, PeanutButter
2	Bread, PeanutButter
3	Bread, Milk, PeanutButter
4	Beer, Bread
5	Beer, Milk

[Bread] ⟶ [PeanutButter] (Sup = 60%, Conf = 75%)

[PeanutButter] ⟶ [Bread] (Sup = 60%, Conf = 100%)

[Beer] ⟶ [Bread] (Sup = 20%, Conf = 50%)

[PeanutButter] ⟶ [Jelly] (Sup = 20%, Conf = 33.33%)

[Jelly] ⟶ [PeanutButter] (Sup = 20%, Conf = 100%)

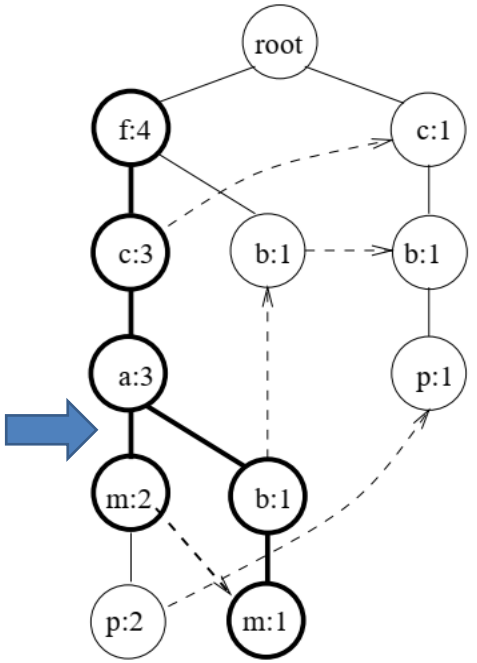
[Jelly] ⟶ [Milk] (Sup = 0%, Conf = 0%)

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIG-MOD. 1993.
Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules." VLDB 1995.

FP-Growth for recommendation

- "FP" stands for Frequent Pattern in a Dataset of transactions
  1. calculate item frequencies and identify frequent items,
  2. a suffix tree (FP-tree) structure to encode transactions, and
  3. frequent itemsets can be extracted from the FP-tree.
- Input: Transaction database
- Intermediate Output: FP-Tree
- Output: {f, c, a → a, m p}, {f, c, a → b, m}

TID	Items Bought	(Ordered) Frequent Items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

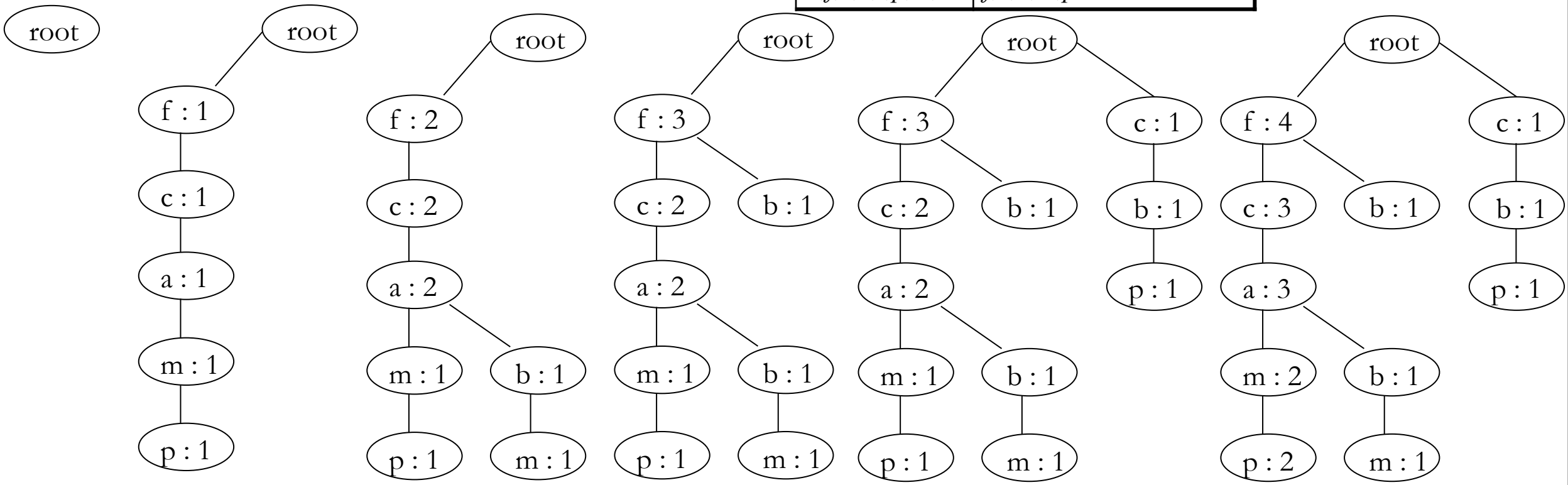


Han Jiawei, Jian Pei, and YiwenYin. "Mining frequent patterns without candidate generation." *ACM SIGMOD Record* 29.2 (2000): 1-12.

FP-Tree Construction

- With *minimum Support threshold* ξ= 3

Items Bought	Ordered Frequent Items
f,a,c,d,g,i,m,p	*f,c,a,m,p*
a,b,c,f,l,m,o	*f,c,a,b,m*
b,f,h,j,o	*f,b*
b,c,k,s,p	*c,b,p*
a,f,c,e,l,p,m,n	*f,c,a,m,p*



Create root

After trans 1
(f,c,a,m,p)

After trans 2
(f,c,a,b,m)

After trans 3
(f,b)

After trans 4
(c,b,p)

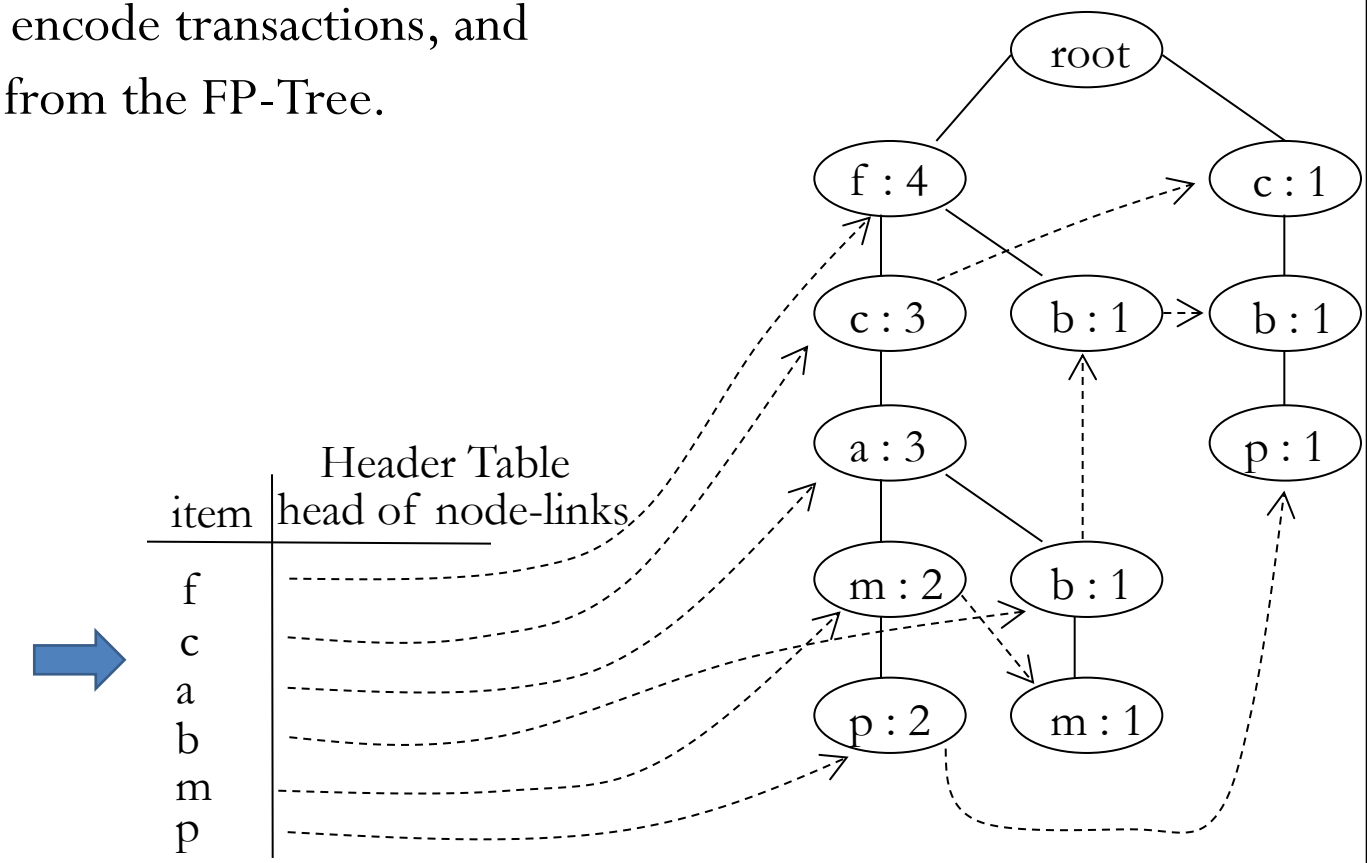
After trans 5
(f,c,a,m,p)

Han, Jiawei, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." ACM SIGMOD Record 29.2 (2000): 1-12.

FP-Growth Rules

- "FP" stands for Frequent Pattern in a Dataset of transactions
  1. calculate item frequencies and identify frequent items,
  2. a suffix tree (FP-Tree) structure to encode transactions, and
  3. frequent itemsets can be extracted from the FP-Tree.
- Input: Transaction database
- Intermediate Output: FP-Tree
- Output example: $\{f \rightarrow c\}$, $\{c \rightarrow a\}$

TID	Items Bought	(Ordered) Frequent Items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p



Han, Jiawei, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." ACM SIGMOD Record 29.2 (2000): 1-12.

PFP: Parallel FP-Growth

- In **Spark ML-Library** (MLLib), a parallel version of FP-growth
  PFP: Parallel FP-Growth
- PFP distributes the work of growing FP-Trees based on the suffixes of transactions.
- More scalable than a single-machine implementation.
- PFP partitions computation, where each machine executes an independent group of mining tasks.
- Google Research

Li, Haoyuan, et al. "PFP: Parallel FP-Growth for query recommendation." *Proceedings of the 2008 ACM Conference on Recommender systems*. 2008.

PFP: Parallel FP-Growth

Map inputs (transactions) key="": value	Sorted transactions (with infrequent items eliminated)	Map outputs (conditional transactions) key: value
f a c d g i m p	f c a m p	p: f c a m m: f c a a: f c c: f
a b c f l m o	f c a b m	m: f c a b b: f c a a: f c c: f
b f h j o	f b	b: f
b c k s p	c b p	p: c b b: c
a f c e l p m n	f c a m p	p: f c a m m: f c a a: f c c: f

Reduce inputs (conditional databases) key: value		Conditional FP−trees
p:	{ f c a m / f c a m / c b }	{(c:3)} | p
m:	{ f c a / f c a / f c a b }	{ (f:3, c:3, a:3) } | m
b:	{ f c a / f / c }	{} | b
a:	{ f c / f c / f c }	{ (f:3, c:3) } | a
c:	{ f / f / f }	{ (f:3) } | c

Li, Haoyuan, et al. "PFP: Parallel FP-Growth for query recommendation." *Proceedings of the 2008 ACM Conference on Recommender systems*. 2008.

PFP: Parallel FP-Growth

- FP-Growth implementation takes the following (hyper-)parameters
  - minSupport: the minimum Support for an itemset to be identified as frequent e.g., if an item appears 3 out of 6 transactions, it has a Support of 3/6=0.5.
  - minConfidence: minimum Confidence for generating Association Rule e.g., if in the transactions itemset X appears 5 times, X and Y co-occur only 3 times, the Confidence for the rule X =>Y is then 3/5 = 0.6.
  - numPartitions: the number of partitions used to distribute the work.
- FP-Growth model provides:
  - freqItemsets: frequent itemsets in the format of DataFrame("items"[Array], "freq"[Long])
  - associationRules: association rules generated with Confidence above minConfidence, in the format of DataFrame("antecedent"[Array], "consequent"[Array], "confidence"[Double]).

https://spark.apache.org/docs/3.3.1/ml-frequent-pattern-mining.html

PFP: Parallel FP-Growth

```
List<Row> data = Arrays.asList(
  RowFactory.create(Arrays.asList("1 2 5".split(" "))),
  RowFactory.create(Arrays.asList("1 2 3 5".split(" "))),
  RowFactory.create(Arrays.asList("1 2".split(" ")))
);
StructType schema = new StructType(new StructField[]{ new StructField(
  "items", new ArrayType(DataTypes.StringType, true), false, Metadata.empty())
});
Dataset<Row> itemsDF = spark.createDataFrame(data, schema);

FPGrowthModel model = new FPGrowth()
  .setItemsCol("items")
  .setMinSupport(0.5)
  .setMinConfidence(0.6)
  .fit(itemsDF);

// Display frequent itemsets.
model.freqItemsets().show();

// Display generated association rules.
model.associationRules().show();

// transform examines the input items against all the association rules and summarize the
// consequents as prediction
model.transform(itemsDF).show();
```

https://spark.apache.org/docs/3.3.1/ml-frequent-pattern-mining.html

תודה רבה
Hebrew

Ευχαριστώ
Greek

Спасибо
Russian

Danke
German

धन्यवादः
Sanskrit

Merci
French

ধন্যবাদ
Bangla

நன்றி
Tamil

شكراً
Arabic

ಧನ್ಯವಾದಗಳು
Kannada

Thank You
English

നന്ദി
Malayalam

多謝
Traditional Chinese

Grazie
Italian

ధన్యవాదాలు
Telugu

આભાર
Gujarati

ਧੰਨਵਾਦ
Punjabi

धन्यवाद
Hindi & Marathi

Gracias
Spanish

多谢
Simplified Chinese

https://sites.google.com/site/animeshchaturvedi07

Obrigado
Portuguese

ありがとうございました
Japanese

ขอบคุณ
Thai

감사합니다
Korean