Exercises Big Data Management

Database Technologies and Information Management (DTIM) group
Universitat Politècnica de Catalunya (BarcelonaTech), Barcelona
February 9, 2024

Contents

1 Big Data Design

1.1 Theoretical questions

1. Briefly explain what "physical independence" is, which is the general position of NOSQL systems on this, and why they take this position.

2. In the framework of the RUM conjecture, name six data structures we saw in the course (not concrete tool implementations) and place them in the corresponding category:

   - Read optimized: B-Tree, Hash, Trie, SkipList
   - Write optimized: LSM, MaSM, PBT, PDT
   - Space optimized: Bloom Filter, Bitmap, Sparse Index

3. Compare a B-tree and an LSM-tree in the context of the **RUM conjecture** (i.e., as an answer to this question, three brief explanations of the form "From the perspective of X, Y-tree is better than Z-tree, because of this and that." are expected).

   (a) R .... B-Tree is better than LSM Tree
   With B-Tree we can directly get location where data is present

   (b) U .... LSM is better than B-Tree because LSM uses an append-only write structure which makes it super-efficient for writes

   (c) M .... LSM is better than B-Tree because in LSM compaction and merging happens which makes it memory efficient

1.2 Problems

Variety
Variability

Variety - different data source, different formats
Variability - data received changes over time, previously no age, now age present

1. Consider:

   a) ['BCN', POPULATION:{VALUE:'2,000,000'}, REGION:{VALUE:'CAT'}]
   b) ['BCN', ALL:{VALUE:'2,000,000;CAT'}]
   c) ['BCN', ALL:{POPULATION:'2,000,000';REGION:'CAT'}]

   Place {a, b, c} in the table:

	Less variable schema	→	**More variable schema**
Less explicit schema			b
↓		c	
More explicit schema	a		

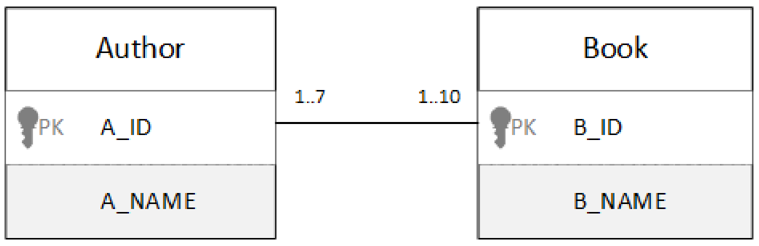
   Name two criteria you would use to choose among them:

   1) How much change is expected in structure of table?
   2)

   Variety of data.
   Variability of the data.

2. Consider the following conceptual schema and propose several design alternatives to translate it to a logical representation (e.g., using JSON notation). Then, explain which is the best alternative and why.

```
{
"author_id" : "  ",
"author_name" : "  ",
"books" : [{
     "book_id" : "  ",
     "book_name" : "  "
    }]
}
```



```
{
"author": [ {"author_id": " ", "author_name" : " }]
}

{
"book": [ {"book_id": " ", "book_name" : " "}],
}

{
"author_book" : [{"author_id": " ", "book_id":" "}]
}
```

```
{
"book_id" : "  ",
"book_name" : "  ",
"authors" : [{
     "author_id" : "  ",
     "author_name" : "  "
    }]
```

Data: a distributed database.
ˆ Information: Internet.
ˆ Query: a distributed database.
ˆ Processing: edge computing.
ˆ RAM: supercomputers.

2 Distributed Data

1. Social Networks - Data is shared
2. Dropbox or Google Drive
3. Google Docs, Google Spreadsheet
4. AWS
5. Video Games
6. Netflix - Distribution of Movies

2.1 Theoretical questions

1. Give five types of data or software resource that can usefully be shared. Give examples of their sharing as it occurs in practice in distributed systems.

Advantage of single server
1. No communication cost
2. Consistency
3. Low Maintenance Cost
4. Easy to recover
5. Security

Disadvantages of single server
1. Scalability
2. Availability
3. Resource Contention

2. Consider the implementation strategies for massively multiplayer online. In particular, what advantages do you see in adopting a single server approach for representing the state of the multiplayer game? What problems can you identify and how might they be resolved?[1]

3. The INFO service manages a potentially very large set of resources, each of which can be accessed by users throughout the Internet by means of a key (a string name). Discuss an approach to the design of the names of the resources that achieves the minimum loss of performance as the number of resources in the service increases. Suggest how the INFO service can be implemented so as to avoid performance bottlenecks when the number of users becomes very large.[1]

Use Hierarchial Structure
Ex: URL
Why hierarchy? To scale out
Hierarchical Naming: Organize resources hierarchically based on their characteristics or categories. This allows for efficient partitioning and searching, reducing the time complexity of resource lookup operations. For example, you could organize resources by category, subcategory, and so on, in a tree-like structure.

4. A server program written in one language (for example, C++) provides the implementation of a BLOB object that is intended to be accessed by clients that may be written in a different language (for example, Java). The client and server computers may have different hardware, but all of them are attached to the Internet. Describe the problems due to heterogeneity of (a) programming languages, and (b) implementations by different developers that need to be solved to make it possible for a client object to invoke a method on the server object.[1]

   Give arguments for and against allowing the client requests to be executed concurrently by the server. In the case that they are executed concurrently, give an example of possible "interference" that can occur between the operations of different clients. Suggest how such interference may be prevented.

Programming languages: Due to the language differences, a common protocol like HTTP or a language independent data format like JSON or XML might be necessary for communication. A method invocation from the Java client to the C++ server will have to be serialized into a standard format that both can understand.
Implementations by different developers: Different developers might have different programming styles, conventions, and assumptions which could introduce bugs and incompatibilities. Standardized interfaces, thorough documentation, and stringent testing practices can help mitigate these issues, follow best coding practices

5. A service is implemented by several servers. Explain why resources might be transferred between them. Would it be satisfactory for clients to multicast all requests to the group of servers as a way of achieving mobility transparency for clients?[1]

Requesting everything to every server is not good - We will end up collapsing the server Find the best server - may be nearest one

Load Balancing -> If one server becomes overloaded with requests or experiences high resource utilization, transferring some resources to other servers can help balance the workload evenly across the server cluster.
Fault Tolerance -> In the event of a server failure or unavailability, resources hosted on that server need to be transferred to other active servers to maintain service continuity. Transferring resources ensures that clients can still access the required resources without disruption.
Geographical Optimization -> In geographically distributed systems, resources may need to be transferred between servers located in different regions to optimize performance or comply with data localization requirements. For example, frequently accessed resources may be replicated or moved closer to the clients to minimize latency.
Scalability -> As the demand for the service grows, additional servers may be added to the system to handle increased workload. Transferring resources between servers allows for dynamic scaling of the service to accommodate changing traffic patterns and user demands.

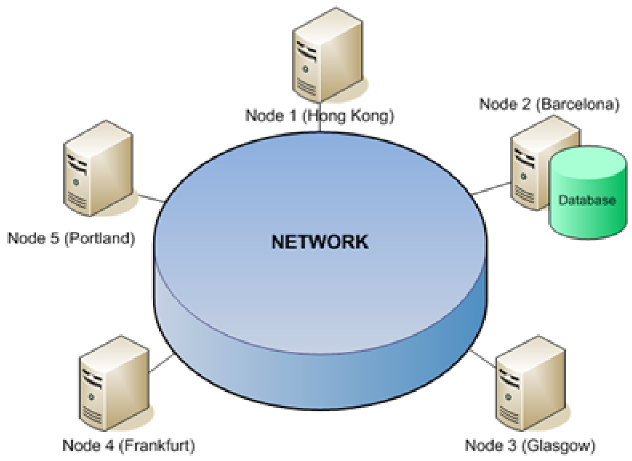
6. Explain what is (a) a distributed system, and (b) a parallel system. Compare both of them (i.e., what has one and not the other and vice-versa).

ds- communicate via network

7. Which kind of database is this according to the distribution of data?

[1]From G. Couloris et al. *Distributed Systems: Concepts and Design*, 5th Ed. Addisson-Wesley, 2012

6. A distributed system is a network of interconnected computers that work together to achieve a common goal. In a distributed system, each computer (or node) operates independently and has its own memory and processing capabilities. These nodes communicate with each other by exchanging messages over the network.
A parallel system is a type of computing system in which multiple processors or processing units work together simultaneously to execute a task.
Concurrency vs. Parallelism: DS emphasize concurrency, enabling multiple tasks to be executed simultaneously across distributed nodes. In contrast, parallel systems focus on parallelism, where tasks are divided and executed concurrently on multiple processing units within a single system.
Communication Overhead: In DS, communication overhead between nodes is typically higher due to the need to exchange messages over the network. In parallel systems, communication overhead is lower as processors often share a common memory or communicate directly with each other.
Fault Tolerance and Scalability: Both distributed and parallel systems can be fault-tolerant and scalable, but they may employ different mechanisms to achieve these goals. DS often rely on replication and redundancy across geographically distributed nodes, while parallel systems may focus on fault tolerance and scalability within a single shared-memory or shared-nothing architecture.



Not Distributed
As data is in one single place

8. Which two kinds of schema information contains the Global Conceptual Schema that does not contain the Local Conceptual Schema in the Extended ANSI-SPARC Architecture for DDBMS?

    1. Fragmentation
    2. Allocation

9. In the context of distributed data management, name the **four big challenges** that need to be carefully considered in the presence of distribution from the tenant/user point of view.

    I. Data Design
    II. Catalog Management
    III. Transaction management
    IV. Query Processing

10. Name the three characteristics of fragmentation that make it correct, and for each of them give an example of fragmentation schema where it is **not fulfilled**.

    (a) Completeness
    (b) Disjointness
    (c) Reconstruction

11. Which is the main problem in having replicas, and which is the innovation introduced by some NOSQL tools to solve it.

    - Problem: Consistency
    - Innovation: Eventual Consistency

12. Given N replicas, let's call R the *ReadConcern* parameter of MongoDB and W the *WriteConcern* (which indicate respectively the number of copies it reads and writes, before confirming the operation to the user); give the equation involving those variables that corresponds to the **eventually consistent** configuration.

    R+W<=N

network latency

13. How might the clocks in two computers that are linked by a local network be synchronized without reference to an external time source? What factors limit the accuracy of the procedure you have described? How could the clocks in a large number of computers connected by the Internet be synchronized? Discuss the accuracy of that procedure.[1]

from a machine send time in receiver add time+ time taken to reach msg there

Synchronizing the clocks of two computers linked by a local network without referring to an external time source can be achieved using various methods. Here's an overview:
1. Synchronization Methods:
Berkeley Algorithm: This involves one computer (master) polling the current time from all other computers (slaves) on the network, calculating the average time, and then setting this average time on all computers.
Cristian's Algorithm: A simpler client-server model where one computer requests the current time from another, and then adjusts its clock based on the received time and the message round-trip time.
2. Synchronizing Over the Internet:
For a large number of computers connected via the Internet, synchronization can be done using the Network Time Protocol (NTP):
NTP: Uses a hierarchy of time servers where computers adjust their clocks based on multiple time references from different servers to improve accuracy and reliability.

14. Query cost refers to the total amount of resources (such as CPU, memory, disk I/O, network I/O) consumed to execute a query, whereas query response time is the time taken to execute a query and return a result.
In centralized systems, query cost and query response time are tightly correlated. As the system is centralized, resource allocation and scheduling are predictable and straightforward, hence the cost directly impacts the response time. Roughly, RT directly prop C.
In distributed systems, the correlation is more complex. Query cost is not the only factor impacting query response time. Network latency, load balancing, data locality, and the consistency model also significantly affect the response time. Thus, a query may have a low cost in terms of resources but may still experience high response time due to network delays or synchronization overhead. In this case, RT directly prop USL(C) (the universal scalability law)

14. What is the difference between query cost and query response time ...

a) in centralized systems (without parallelism)?

b) in distributed systems (inherently parallel)?

15. Name the two factors that make impossible having linear scalability according to the Universal Scalability Law.

(a) Contention: fraction of the algorithm that is not parallelizable

(b) Coherency: how much the different parallel units require communication

Programs are not fully parallelizable.
There are communication costs.

2.2 Problems

1. Briefly explain (a) which fragmentation strategy has been applied for the tables below and whether this fragmentation strategy is (b) complete, (c) disjoint and (d) allows to reconstruct the global relations (if so, (e) indicate the operation).

Global Relations

```
Kids(kidId, name, address, age)
Toys(toyId, name, price)
Requests(kidId, toyId, willingness)

Note that requests(kidId) is a foreign key to kids(kidId) and
similarly, requests(toyId) refers to toys(toyId).
```

Fragments

K1= Kids[kidId, name]

K2= Kids[kidId, address, age]

Vertical
Complete
Disjoint - Id is ignored
Reconstructable - Join

T1= Toys(price $\geq$ 150)

T2= Toys(price $<$ 150)

Horizontal
Complete (if no null values)
Disjoint
Reconstructable - Union (if complete)

R1 = Requests $\ltimes$ T1

R2 = Requests $\ltimes$ T2

Derived Horizontal Fragmentation
Complete (if no null values)
Disjoint
If complete - Reconstructable -> Union

2. You are a customer using an e-commernce based on heavy replication (e.g., Amazon):

Eventually Consistent
Lazy Replication

a) Show a database replication strategy (e.g., sketch it) where:

i. You buy an item, but this item does not appear in your shopping cart.

ii. You reload the page: the item appears.

⇒ What happened?

N=3
W=1
R=1

b) Show a database replication strategy (e.g., skecth it) where:

i. You delete an item from your command, and add another one: the shopping cart shows both items.

⇒ What happened? Will the situation change if you reload the page?

Eventually Consistent
Lazy Replication

3. Consider the following architectures and answer the questions:[2]

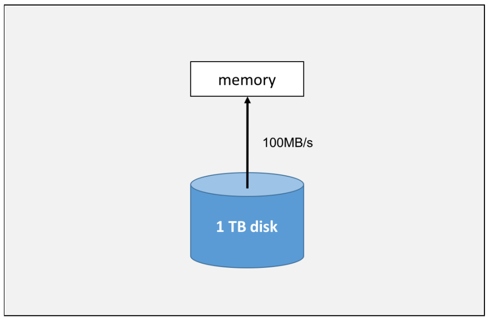


Figure 1: Centralized architecture

Type	**Latency**	**Bandwidth**
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.)	At best 100 MB/s

a) How long would it take (i.e., response time) to read 1TB with sequential access (Figure 1)? (in secs) (1 TB/100 MB)+5*10-3

b) How long would a single random access (i.e., reading one tuple, of for example 100 bytes, through an index) take (i.e., response time), assuming we already have the physical address? (in secs)

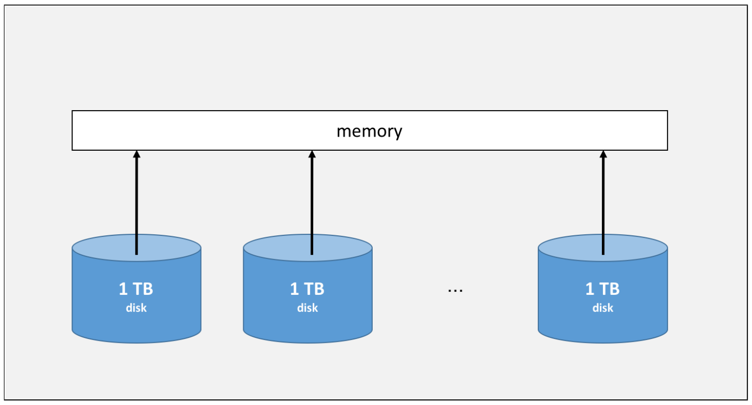


Figure 2: Shared-memory architecture

Type	**Latency**	**Bandwidth**
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.)	At best 100 MB/s

c) How long would it take (i.e., response time) to read 1TB with parallel access (Figure 2)? Assume 100 disks (i.e., 100 replicas of the whole data) on the same machine with shared-memory and infinite CPU capacity.

d) How long would a single random access (i.e., reading one tuple, of 100 bytes, through an index) take (i.e., response time), assuming we already have the physical address? (in secs)

[2]From S. Abiteboul et al. *Web Data Management*. Cambridge Press, 2011.

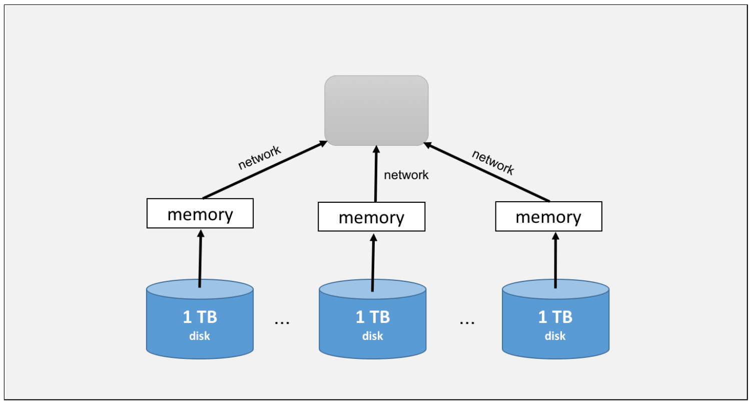


Figure 3: Shared-nothing architecture

Type	**Latency**	**Bandwidth**
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.)	At best 100 MB/s
LAN	$\approx 1 - 2 \times 10^{-3}$s (1-2 millisec.)	$\approx$ 1 GB/s (single rack) $\approx$ 10MB/s (switched)
Internet	Highly variable (typically 10-100ms)	Highly variable (typically a few MB/s, e.g., 10MB/s)

Note 1: It is approx. one order of magnitude faster to exchange main memory data between 2 machines in a data center, that to read on the disk.

Note 2: Exchanging through the Internet is slow and unreliable with respect to LANs.

e) How long would it take (i.e., response time) to read 1TB with distributed access (Figure 3)? Assume 100 shared-nothing machines (with all data replicated in each of them) in a star-shape LAN in a single rack where all data is sent to the center of the star in only one network hop.

f) How long would it take (i.e., response time) to read 1TB with distributed access (Figure 3)? Assume 100 shared-nothing machines (with all data replicated in each of them) in a star-shape cluster of machines connected through the Internet where all data is sent to the center of the star in only one network hop.

g) How long would a single random access (i.e., reading one tuple, of 100 bytes, through an index) take (i.e., response time) in the case of a LAN, assuming we already have the physical address? (in secs)

h) How long would a single random access (i.e., reading one tuple, of 100 bytes, through an index) take (i.e., response time) in the case of an Internet connection, assuming we already have the physical address? (in secs)

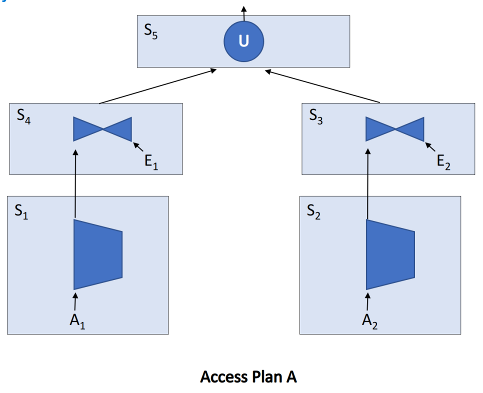
4. What are the main differences between these two distributed access plans? Under which assumptions is one or the other better?

```
SELECT *
FROM employee e, assignedTo a
WHERE e.#emp = a.#emp AND a.responsability = 'manager'
```

1. In plan A, firstly, data is reduced by means of selections. Then, they are joined, increasing the data. Finally, these joined tables are transferred through network.
2. In plan B, the plain tables are transferred to one node, and then they are combined, selected and joined. All in one machine.

There are several considerations that need to be taken into account to decide between these two approaches:
In plan B, if the individual tables are large, it is possible that one node does not have enough capacity to store them. In these cases, we can only go for A.

If that's not the case, everything would depend on how big the joined tables in plan A are compared to the raw tables, since the transfer cost is probably the dominating cost here. If the joined tables are smaller than the individual tables, then plan A would be better, else B.



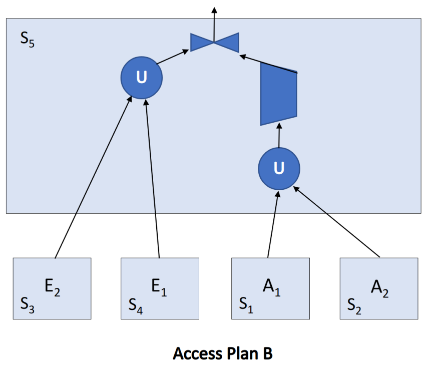


Figure 4: Distributed Access Plans

```
Employee(#emp, empName, degree)
S4:  E1 = Employee(#emp ≤ 'E3')
S3:  E2 = Employee(#emp > 'E3')

AssignedTo(#emp, #proj, responsability, fullTime)
FK: #emp references Employee
S1:  A1 = AssignedTo(#emp ≤ 'E3')
S2:  A2 = AssignedTo(#emp > 'E3')
```

5. Compute the fragment query (data location stage) for the database setting and query below and find in how many ways we can assign the operations to the different sites.

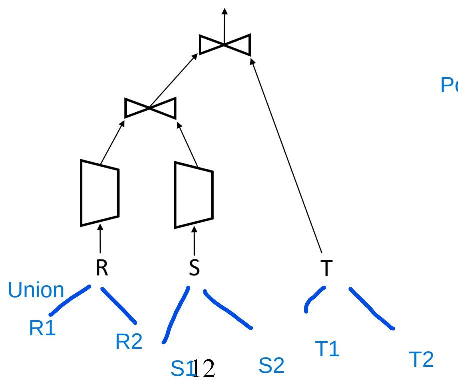
⇒ **Database setting:**

- A distributed database with 5 sites (i.e., database nodes): S_1, S_2, S_3, S_4 and S_5.
- 3 relations in the database R, S and T
- Each relation is horizontally fragmented in two fragments (we refer to them by the name of the relation and a subindex, for example: R_1, R_2). You can consider them to be correct (i.e., complete, disjoint and reconstructible).
- Each fragment is replicated at all sites.
- We have the following query: $Q_1 = \sigma(R) \bowtie \sigma(S) \bowtie T$

⇒ **Process tree of the query:**

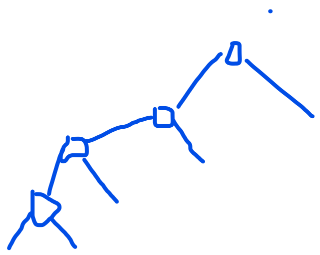
Notice that since each relation is horizontally fragmented into two fragments, we need to perform the union between those to get the relations. For example, R = R1 U R2.
Therefore, the tree has 3 more nodes for each relation (one for each segment, and one for the union).

Thus, the tree is formed of 13 nodes, and there are 5 possible sites in which each node can execute.
This means that there is a total of 5^13 possible arrangements.



Possibility = 5^13

6. Consider a left-deep process tree corresponding to a query, where each internal node is a join, and every leaf a data source (e.g., relational table). Knowing that the tree contains 9 nodes (including leaves), the system has as much parallelism capacity as needed to run all the joins in pipelining mode (no other kind of parallelism is available), which is the occupancy if the overall cost of the serial query is 4 seconds? Explicit any assumption you need to make.



Tree is as shown in the left.
Then, there are 4 operators, so the occupancy is
O = T/ N = 4s/4op = 1s/op

There are no stalls since we assumed an indefinite parallelism capacity for pipelining.

Q) A server process maintains a shared information object. Give arguments for and against allowing the client requests to be executed concurrently by the server. In the case that they are executed concurrently, give an example of possible 'interference' that can occur between the operations of different clients. Suggest how such interference may be prevented.
-> Allowing concurrent client requests can increase the system's throughput but can also lead to data inconsistencies or conflicts if not properly managed. For example, if two clients try to modify the same data at the same time, one client's changes could overwrite the other's. This interference can be prevented using concurrency control techniques such as locking (where a resource is locked when a client is modifying it and other clients have to wait), or optimistic concurrency control (where changes are checked for conflicts before being committed).

Randomly allocated
Large file is divided into smaller chunks (distribution unit)
Why? Don't want data to be skewed - want uniform distribution of data
Chunk size is fixed - some chunk might not be filled
Has balancer to later rebalance the distribution of chunks

3 Distributed File System

3.1 Theoretical questions

1. How does HDFS decide where to place the different chunks of a file?

3.2 Problems

1. Given a table with ten blocks ($B = 10$) and 17 tuples per block ($R = 17$) for a total of 170 tuples, compare the cost of reading the whole table sequentially, against the cost of accessing seven random (potentially repeated) tuples in an unknown order. Assume that seek time is $12ms$, average rotation time is $3ms$ and transferring one block is $0.03ms$. You should consider which is the probability of two tuples accessed consecutively are actually in the same block, but ignore the presence of any cache or buffer pool mechanism.

2. Consider a table stored in HDFS with the following characteristics:

size(T)	=	256 MB	Size of table
$\|T\|$	=	64	Number of rows
size(row)	=	4 MB	
cols(T)	=	4	Number of columns
size(cell)	=	1 MB	
size(Header)	=	0	
size(Footer)	=	0	

a) Given the configuration parameters, how much space would you need to store it in a horizontal layout (i.e., Avro)?

Size(MetaRow)	=	0.1MB
Size(MetaBody)	=	0.1MB

b) Given the configuration parameters, how much space would you need to store it in a hybrid layout (i.e., Parquet)?

Size(MetaCol)	=	0.2MB
Size(MetaRowGroup)	=	0.5MB
Size(RowGroup)	=	8MB

c) Given the configuration parameters, how much data would you retrieve in a hybrid layout (i.e., Parquet) to project two columns?

Size(MetaCol)	=	0.2MB
Size(MetaRowGroup)	=	0.5MB
Size(RowGroup)	=	8MB

d) Given the configuration parameters, how much data would you need to retrieve it in a hybrid layout (i.e., Parquet) to select one row (e.g., given its key) if the table is not sorted?

Size(MetaCol)	= 0.2MB
Size(MetaRowGroup)	= 0.5MB
Size(RowGroup)	= 8MB

3. Given a file with C chunks and N nodes in the cluster, find the probability that HDFS uses all the nodes in case $C \geq N$, by using Stirling numbers of the second kind.[3]

4. Consider an HDFS cluster with 100 data nodes, without replication. If I upload a file with 10 chunks and 10 disk blocks each, answer the following questions and briefly justify your answer:

   a) Which is the maximum number of machines containing data?

   Since there is no replication, and the unit of distribution is the chunk, then at most 10 nodes can contain data.

   b) Which is the probability of the maximum number of machines actually contain data?

   The probability that all chunks end up in different machines is
   =(100/100)*(99/100)*(98/100)*(97/100)*(96/100)*(95/100)*(94/100)*(93/100)*(92/100)*(91/100) = 0.6282

5. Given a file with $3.2GB$ of raw data stored in an HDFS cluster of 50 machines, and containing $16 \cdot 10^5$ rows in a **Parquet file**; consider you have a query over an attribute "$A =$ constant" and this attribute contains only 100 different and equiprobable values. Assuming any kind of compression has been disabled, explicit any assumption you need to make and give the amount of raw data (i.e., do not count metadata) it would need to fetch from disk.

   40 chunks to be fetched

   - Replication factor: 3 (default)
   - Chunk size: $128MB$ (default)
   - Rowgroup size: $32MB$

   There are 3.2GB/ 32MB =100 RowGroups in the cluster.
   Since the file contains 16 × 10^5 rows, each rowgroup contains 16 ×10^5/ 100 =1600 rows.
   Since attribute A contains 100 different and equiprobable values, then we can consider that 16 rows will be fetched from each rowgroup.

   In a chunk, there are 4 rowgroups

6. Consider a cluster of ten worker machines and a single coordinator, which contains a sequence file of 128MB stored in HDFS with an OS block size of 32KB in all machines and a chunk size of 64MB. On the event of a client retrieving that file, give the **number of control messages** (do not consider data messages) that will travel the network in the case of (a) a client cache miss and in the case of (b) a client cache hit. Briefly justify both numbers.
   3 2

Client cache miss:
In this case, when a client requests a file, the client first contacts the NameNode to get the location of the first block. The NameNode contacts the nodes in which each chunk is located, and then they answer the client. Therefore, there are 3 control messages, one from client to namenode, and two from namenode to datanodes.

Client cache hit
In this case, the client knows the locations of the servers, so it asks directly them, accounting for just 2 control messages.

Q) Pg 17 Lorencio Notes

[3] https://en.wikipedia.org/wiki/Stirling_numbers_of_the_second_kind

4 Key-Value Stores

Linear hash adapts as the system grows, and there are two functions at once.
Consistent hash consists of a predefined static hash function.

4.1 Theoretical questions

Hash function changes Single has function

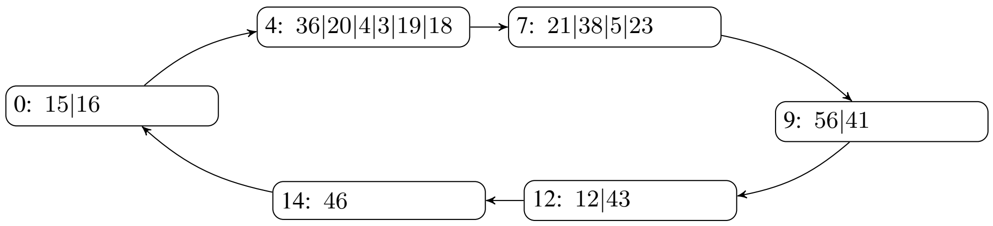
1. Which is the main difference between the hash functions used in the linear hash and consistent hash algorithms?

2. With respect to distributed systems, explain what is a distributed hash table (DHT), and provide a brief description of how consistent hashing guarantees balancing keys when adding new servers.

A DHT is a hash table that is divided into buckets, having these buckets distributed.
Consistent hashing guarantees balancing by redistributing the keys at the moment of adding new servers, and relies on the big numbers and the assumption of well-constructed hash functions.

4.2 Problems

1. Let's assume we have a *Consistent hash* with $D = 16$, and the hash function is simply the module of the IP address or the key, and suppose the current state of the consistent hash is (position_in_the_ring:key|key|...):



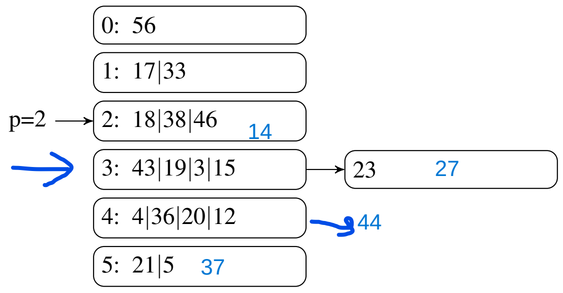
   a) What happens when we insert objects 30 and 58? Draw the result.

   14 goes to 12

   b) What happens in the structure when we register a new server with IP address 37? Draw the result.

   server 5 is added between 4 and 7
   21, 5 goes to this server

2. Let's suppose we have a *Linear Hash* and the hash function is simply the module of the key, the capacity of a bucket is only four entries, and current state of the linear hash is (bucketID: key|key|...):



6: 38|46|14

   a) What happens in the structure when we insert keys 14, 27, 37, and 44? Draw the result.

3. Let's suppose that, we have an *LSM Tree* that reached the threshold to consider the `MemStore` is full, and it contains four entries with format $[key, value, timestamp]$ needing ten characters each. The content of the different structures is:

   - MemStore: $[1, v, t50]$, $[15, v, t49]$, $[17, v, t47]$, $[29, v, t48]$
   - Commit Log: $[17, v, t47]$, $[29, v, t48]$, $[15, v, t49]$, $[1, v, t50]$

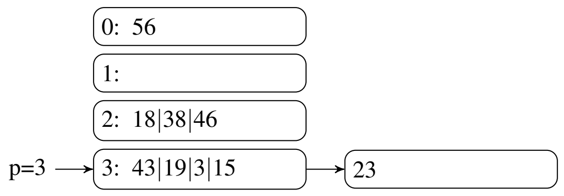
- $\text{SSTable}_{\text{Data}}$: $[13, v, t23], [25, v, t17], [35, v, t40], [59, v, t38]$
- $\text{SSTable}_{\text{Index}}$: $[13, 0], [25, 30], [35, 60], [59, 90]$

Assuming that the minimum size of an `SSTable` is 120 characters and on having two `SSTables` a minor compactation is automatically triggered, explicit the content of all structures once the compactation is done.

- MemStore: Empty
- Commit Log: Empty
- $\text{SSTable}_{\text{Data}}$: [1,v,t50], [13,v,t23], [15,v,t49], [17,v,t47], [25,v,t17], [29,v,t48], [35,v,t40], [59,v,t3
- $\text{SSTable}_{\text{Index}}$: [1,0],[13,30],[15,60],[17,90],[25,120],[29,150],[35,180],[59,210]

sorted by key
not by timestamp

4. Briefly explain what is wrong in this linear hash structure, or if you think it is right, explicitly say so.

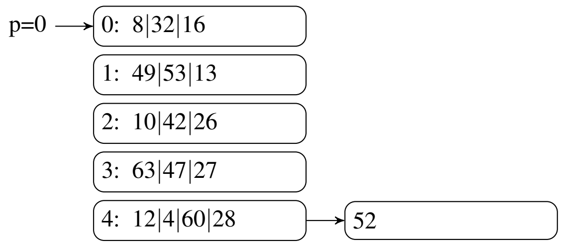


if pointer is in 3,
0,1,2 must have already been splited
so there should have been bucket after 3

Pointer cannot be anywhere

In this case, n = 2, and it is not possible to have the pointer pointing to a bucket between 2^n and 2^n+1, because the design of the algorithm makes this impossible. When 2^n + 1 is reached, the pointer goes back to the first bucket and the process starts all over.

5. Given the **linear hash** underneath with $f(x) = x$ (i.e., we directly apply the module to the keys), and a capacity of four keys per bucket, indicate if it corresponds to a state **valid or not**. If valid, give a possible insertion order leading to it. If not, clearly explain why.



Invalid

6. Suppose you have a hash function whose range has size 100 (i.e., D=100), and a Consistent Hash structure with 5 machines (M1...5) whose identifiers map to values $h(M1) = 0$, $h(M2) = 20$, $h(M3) = 40$, $h(M4) = 60$, $h(M5) = 80$. What happens if you have an object mapped to value $h(O) = 90$?

Goes to machine 1

7. Given an empty **Consistent Hash** with $h(x) = x\%32$ (i.e., we directly take module 32 to both the keys and the bucket IDs), and unlimited capacity in each bucket, consider you have a cluster of four machines with IDs 19, 22, 75, 92, and draw the result of inserting the following keys in the given order: 12, 4, 10, 49, 42, 60, 63, 53, 47, 27, 26, 28, 13, 52.

8. Suppose you implement a system to store images in hundreds of machines with thousands of users using HBase with a single column-family. These images taken at time VT belong to a person P who tags each with a single subject S (e.g., family, friends, etc.) and are concurrently uploaded into the system at time TT in personal batches containing multiple pictures of different subjects taken at different times. Each person can then retrieve all his/her pictures of one single subject that were taken after a given time. Precisely define the key you would use (which cannot be a hash) if you exclusively prioritize (i.e., do not consider any other criteria)...

(a) Load balancing on ingestion:

   ⇒ Assumptions made:

(b) Load balancing on querying:

   ⇒ Assumptions made:

(c) I/O cost (i.e., minimum blocks flushed) on ingestion:

   ⇒ Assumptions made:

(d) I/O cost (i.e., minimum blocks retrieved) on querying:

   ⇒ Assumptions made:

9. Given an empty **linear hash** with $f(x) = x$ (i.e., we directly apply the module to the keys), and a capacity of four keys per bucket, draw the result of inserting the following keys in the given order: 12, 4, 10, 49, 42, 60, 63, 53, 47, 27, 26, 28, 13, 52.

5 Document Stores

5.1 Problems

1. Optimize the performance of the following document design:

```
{"_id": 123,
 "name": "Alberto"
 "address": {
  "street": "Jordi Girona",
  "city": "Barcelona",
  "zip_code": "08034"},
 "telephone1": "93 4137889",
 "telephone2": "93 4137840",
 "telephone3": "123456789"
}
```

Store telephone in array
No need to nest address
can also add single address

2. Analyze (i.e., briefly give **pros and cons**) the following **JSON design** compared to other equivalent JSON designs from the three perspectives:

```
"BID-PRODUCT":{
  "B_ID": int(4), "B_PRICE": int(4), "U_ID": int(4),
  "PRODUCT": {
    "P_ID": int(4), "P_INFO": varchar(100)
}}
"PRODUCT-SELLER-REGION": {
  "P_ID": int(4), "P_INFO": varchar(100),
  "USER": {
    "U_ID": int(4), "U_F_NAME": varchar(20),
    "REGION": {
      "R_ID": int(4), "R_NAME": varchar(10)
}}}
"PRODUCT-COMMENTS": {
  "P_ID": int(4), "P_INFO": varchar(100),
  "COMMENTS": [{
    "C_ID": int(4), "C_TITLE": varchar(20), "U_ID": int(4)
}]}
```

Read (a.k.a. Query):
Pros: retrieving the information regarding a product is very easy, like the sellers of a product. Also, retrieving all the comments of a product is easy.
Cons: retrieving all products that a seller offers requires many joins, as well as obtaining all bids to a product and even worse is to and all products sold in a given region.

Update:
Pros: if we want to update the sellers of a particular product, it is fast and easy. Same if we want to update the comments of a particular product or the product of a bid.
Cons:
If we want to update the region of a seller, we would need to traverse all product to see if the seller sells that product, and then update the region. This is very costly and ineffcient.
Also, if we want to update a product description, we would need to change it in the three places that it appears.

Memory:
Pros: Having product as a central concept makes BID-PRODUCT and PRODUCT-COMMENTS able to be stored efficiently, since we just need to provide the ID of the related product.
Comments are stored as list so space efficient instead of storing as seperate comment
Cons: P_INFO is stored repeatedly. Not only this, but we are storing the USERS many times, and the REGIONS even more times.

(a) Read (a.k.a. Query)
  i. Pros:
  ii. Cons:

(b) Update
  i. Pros:
  ii. Cons:

(c) Memory (a.k.a. Space)
  i. Pros:
  ii. Cons:

3. Optimize the performance of the following pipe:

Start with
1. match
2. Add new projection here
3. Sort
4. Project

```
db.partsupp.aggregate([
 {"$sort": {"n_name": 1, "s_name": 1, "p_partkey": 1}},
 {"$match": {"part.p_size": 5},
 {"$project": {
   "supplier.s_name": 1,
   "supplier.s_phone": 1,
   "supplier.s_comment": 1,
   "_id": 0
   }}
])
```

4. Assume a MongoDB collection of JSON documents following the structure shown below. Indicate a sequence of stage operators that can be used in a pipeline to compute the top five departments in terms of number of employees such that its location has at least one `performance_review` greater than 7. In the example below, since both locations have performance reviews greater than 7, `manufacturing` has a total of 90 employees, while `billing` and `sales` have 80 and 40 employees, respectively. You need only provide the names of the stage operators and the sequence in which they are applied. You should not use the `mapreduce` feature of MongoDB but rather the specific stage operators provided by MongoDB. You do not need to explicitly handle the case of ties (i.e., in the case of ties, you may return any of the tied documents to complete the top five).

```
{ "location": "Barcelona",
  "budget": 3.000.000,
  "departments": [ { "name": "sales", "employees": 40 }, { "name": "manufacturing",
  "employees": 75 } ],
  "performance_reviews": [ 7, 6, 3 ]
},
{ "location": "Brussels",
  "budget": 5.000.000,
  "departments": [ { "name": "manufacturing", "employees": 15 }, { "name": "billing
  ", "employees": 80 } ],
  "performance_reviews": [ 6, 8, 7 ]
}
```

$match: Filter documents where at least one performance review is greater than 7.
$unwind: Deconstruct the departments array so that each department becomes a separate document.
$group: Group documents by department name, calculating the total number of employees for each department.
$sort: Sort departments by the total number of employees in descending order.
$limit: Limit the output to the top five departments.

5. Imagine you want to design a system to maintain data about citizens and doubt whether to use HBase or MongoDB. Precisely, for each citizen, it will store: their personal data (with pID), their city data (with cID) and their employment data. We also know the workload (i.e., queries and frequency of execution) is as follows:

- Q_1: Average salary in Barcelona (50% frequency) – information obtained from the set of city and employment data.
- Q_2: Average weight for young people (less than 18 years old) (45% frequency) – information obtained from the set of personal data.
- Q_3: Number of VIPs (5% frequency) – information obtained from the set of personal data.

⇒ Discuss your choice of technology and data model, without pre-computing the results of the queries. Clearly specify the structure of the data (i.e., tables/collections, keys, values, etc.), trade-offs and assumptions made.

HBase-range queries
Average, Aggregate - MongoDB
filter - mongodb support sec index
Scability needed HBase

6. Assume you have a MongoDB collection which occupies 6 chunks **evenly distributed** in 3 shards (i.e., 2 chunks per shard). Being the document ID also the shard key, the chunk of a document is determined **by**

means of a hash function. Assuming that accessing one document takes one time unit (existing indexes are used at no cost) and we have 6,000 documents in the collection, k of which have value "YYY" for attribute "other", how many time units would take the following operations[4]:

a) $FindOne(\{\_id : "XXX"\})$ Key is always indexed 1

b) $Find(\{\_id : \{\$in : [1, .., 3000]\}\})$, being [1,..,6000] the range of existing IDs. 1000

c) $Find(\{other : "YYY"\})$, being the attribute indexed. k/3

d) $Find(\{other : "YYY"\})$, being the attribute NOT indexed. 2000

7. Assume you have a MongoDB collection which occupies 6 chunks **UNevenly distributed** in 3 shards (i.e., 1, 2 and 3 chunks per shard respectively). Being the document Id also the shard key, the chunk of a document is determined **by means of a hash function**. Assuming that accessing one document takes one time unit (existing indexes are used at no cost) and we have 6,000 documents in the collection, k of which have value "YYY" for attribute "other", how many time units would take the following operations[5]:

e) $FindOne(\{\_id : "XXX"\})$ 1

f) $Find(\{\_id : \{\$in : [1, .., 3000]\}\})$, being [1,..,6000] the range of existing IDs. 1500

g) $Find(\{other : "YYY"\})$, being the attribute indexed. 3k/6

h) $Find(\{other : "YYY"\})$, being the attribute NOT indexed. 3000

8. Assume you have a MongoDB collection which occupies 6 chunks and is **evenly distributed** in 3 shards (i.e., 2 chunks per shard) . Being the document Id also the shard key, the chunk of a document is determined **by range**. Assuming that accessing one document takes one time unit (existing indexes are used at no cost) and we have 6,000 documents in the collection, k of which have value "YYY" for attribute "other", how many time units would take the following operations[6]:

i) $FindOne(\{\_id : "XXX"\})$ 1

j) $Find(\{\_id : \{\$in : [1, .., 3000]\}\})$, being [1,..,6000] the range of existing IDs. 2000

k) $Find(\{other : "YYY"\})$, being the attribute indexed. k/3

l) $Find(\{other : "YYY"\})$, being the attribute NOT indexed. 2000

[4] As typically in RDBMS optimizers, assume uniform distribution of values and statistical independence between pairs of attributes.
[5] As typically in RDBMS optimizers, assume uniform distribution of values and statistical independence between pairs of attributes.
[6] As typically in RDBMS optimizers, assume uniform distribution of values and statistical independence between pairs of attributes.

6 New Relational Architecture

6.1 Theoretical questions

data can be stored in n memory places
If high associativity, facilitates replacement but affects searching

1. Briefly explain the concept of associativity in the context of memory usage. Identify the benefit of high (respectively low) associativity.

6.2 Problems

1. Assume you have a SanssouciuDB table T stored **row-wise**, which ocupies 300 blocks and contains tuples with three variable length attributes [A, B, C] (underlined attribute is declared to be the primary key of the table). Assume there is **no index** and give the **average cost** of each query assuming accessing one block is one second.

   - SELECT A,B,C FROM T WHERE A=x; (being x a constant)

     150.5
     best case =1
     worst case =100
     so average

   - SELECT SUM(B) FROM T; 300

2. Assume you have a SanssouciuDB table T stored **column-wise**, which ocupies 300 blocks and contains tuples with three attributes [A, B, C] (underlined attribute is declared to be the primary key of the table). Assume there is **no index**, storage of each attribute uses exactly the same space after compression (i.e., 100 blocks), and run length encoding has been applied for non-key attributes storing ending row position per run. Give the **average cost** of each query assuming accessing one block is one second and explicit any other assumption you make.

   - SELECT A,B,C FROM T WHERE A=x; (being x a constant) 52

   - SELECT SUM(B) FROM T; 100

3. Assume that there is a table T with attributes [A,B,C] (the underlined attribute is the primary key of the table), which occupies 300 disk blocks of 8KB each, with 100bytes per row. All three attributes require the same space (even in case of compression). Supposing that there is not any index, which would be the **minimum amount of memory required to process** (do not consider the memory required to store the final output of the query) the following query in either row storage or column storage (briefly **justify** your answer and explicit any assumption you make).

   SELECT A FROM T WHERE B='x' AND C='y';

   - Row storage 1 block

   - Column storage 3 block

4. Without considering compression and assuming a disk block size of 8KB, is there any case where a query retrieving all tuples, but only half of the equally-sized attributes of a relational table performs better in **row storage** without any kind of vertical partitioning than in columnar storage? Numerically justify your answer.

no
B
B/2

5. Represent the given column with dictionary and run-length encoding storing end row position.

A
A
B
B
B
C
A
A

Table

1 A
2 B
3 C

Dictionary

1
4
5
7

End Row

1
2
3
1

Code

6. Given the two columns of a table represented with run-length encoding using dictionary, give the position of the row(s) that fulfill the predicate "Charlie AND Beta". Briefly explain how a column store would obtain them.

Bravo
Charlie

Dictionary

# values	**Dictionary positions**
2	1
3	0
3	2

Alpha
Beta

Dictionary

# values	**Dictionary positions**
1	1
5	0
2	2

The first two values in the first column are Charlie, and no more Charlie's are encoutered.
In the second column, then, we have to check these two positions. Since Beta is in the first position, and is encountered only once, then this one is the only solution: row 0

7 Distributed Processing Frameworks

7.1 MapReduce in use

Hybrid
first phase - map phase + half of shuffle phase- query shipping
second phase - second half of shuffle + reduce phase - data shipping

7.1.1 Theoretical questions

1. Classify a MapReduce system as either query-shipping (i.e., the evaluation of the query is delegated to the site where the data are stored), data-shipping (i.e., the data are moved from the stored site to the site executing the query) or hybrid (i.e., both query- and data-shipping). Clearly justify your answer.

7.1.2 Problems

1. Consider the following implementation of the Cartesian Product with MapReduce ($\oplus$ stands for concatenation) and answer the questions accordingly.

$$T \times S \Longrightarrow \begin{cases} \text{map(key k, value v)} \mapsto & \\ [(h_T(k) \mod D, k \oplus v)] & \text{if input}(k \oplus v) = T \\ [(0, k \oplus v, ..., (D-1, k \oplus v)] & \text{if input}(k \oplus v) = S \\ \text{reduce(key ik, vset ivs)} \mapsto & \\ [crossproduct(T_{ik}, S)] & \\ T_{ik} = \{iv | iv \in ivs \wedge input(iv)T\}, & \\ S = \{iv | iv \in ivs \wedge input(iv)S\}] & \end{cases}$$

   (a) Which is the relationship of D with parallelism and scalability, if any?

   After sometime - not that useful
   startup time more required
   each reduce - not sufficient work to do

   D is directly proportional to the level of parallelism in the MapReduce computation.
   D stands for the number of reducers, and increasing D allows for more tasks to be executed in parallel, enhancing the computation's scalability.

   (b) Which is the optimal value for D?

   The optimal value of D depends on the specific usecase, the size and characteristics of the data, and the resources available in the cluster.
   In general, it's best to have D sets so that each reducer has a manageable amount of data to process and the work load is evenly distributed across all reducers.
   A reasonable value for D could be 2×Nmachines, so that every machine can be activated, and if some are faster, they can still keep processing

   2 times
   or 0.95 in notes

2. Provide the MapReduce pseudo-code implementation of the following Relational operators. You can use "$\oplus$" symbol for concatenation, "$prj_{a_{i_1}...a_{i_n}}(t)$" to get attributes "$a_{i_1}...a_{i_n}$" in t, "$crossproduct(S_1, S_2)$" to perform the cross product of two sets of rows, "$input(t)$" to know if t comes from either T or S, and assume the "key" parameter contains the PK of the table and the "value" one all the others.

   - Aggregation ($\gamma_{A,sum(B)}(T)$)
   - Selection ($\sigma_{A='x'}(T)$)
   - Join ($T \bowtie_{T.A=S.B} S$)
   - Union (T ∪ S)
   - Difference (T \ S)
   - Intersection (T ∩ S)

3. In relational algebra, the antijoin operator ($\triangleright$) is defined as the complement of the semijoin on the primary keys (PKs). Formally, assuming A and B are the PKs of R and S, respectively, then $R \triangleright S = R \setminus R \ltimes_{A=B} S$. Provide the MapReduce pseudo-code implementation of the antijoin operator. Assume the existence of the operator $\oplus$ to concatenate strings, $prj_{att}(s)$ to project attribute att from the tuple s, and $input(s)$ to decide the origin (i.e., R or S) from a tuple s.

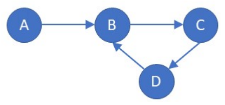
4. **Modify** the following **MapReduce code** in Python so that given a file with key-value pairs representing the edges of a graph[7] (i.e., the key is the label of the first node and the value is the label of the second), it identifies all pairs of nodes connected by a path of length two (if two nodes are connected multiple times, they will be repeated in the output). Do not need to optimize it in any way, just make it work as expected.

```
def map(k,v):
        return [(k,(k,v))]        [ (k ,(k,v)) ,(v ,(k,v)) ]

def reduce(k, lv):
        outgoing = []
        incoming = []
        for v in lv:
                if k == v[0]:
                        outgoing.append(v[1])
                elif k == v[1]:
                        incoming.append(v[0])
        retValue = []
        for o in outgoing:
                for i in incoming:
                        retValue.append(i+"-2->"+o)
        return retValue
```

Example of Input

Key	**Value**
A	B
B	C
C	D
D	B



Expected output

```
A -2-> C
D -2-> C
B -2-> D
C -2-> B
```

7.2 MapReduce internals

7.2.1 Problems

1. Assume the following MapReduce program:

```
public void map(LongWritable key, Text value) {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        write(new Text(tokenizer.nextToken()), new IntWritable(1));
    }
}
public void reduce(Text key, Iterable<IntWritable> values) {
    int sum = 0;
    for (IntWritable val : values) {
       sum += val.get();
    }
    write(key, new IntWritable(sum));
}
```

Consider the following data set:

- Block0: "a b b a c | c d c e a"
- Block1: "a b d d a | b b c c f"

Simulate the execution of the MapReduce code given the following configuration:

- The **map** and **reduce functions** are those of the wordcount
  - The **combine** function shares the implementation of the reduce
- One **Split** is one block

[7]Without reflexive edges like (A,A).

- The "|" divides the **records** inside each block
  - We have two records per block
- Hadoop is configured with the parameter *dfs.replication=1*
- We can keep four pairs [key,value] per **spill**
- We have two **mappers** and two **reducers**
  - $Machine0$, contains block0, runs mapper0 and reducer0
  - $Machine1$, contains block1, runs mapper1 and reducer1
- The hash function used to shuffle data to the reducers uses the correspondence:
  - $\{b,d,f\} \rightarrow 0$
  - $\{a,c,e\} \rightarrow 1$

Fill the gaps in each step (numbers correspond to the phase in the MapReduce algorithm):

1) $Machine0$ contains · 1 · block(s).
   $Machine1$ contains · 1 · block(s).
2) We keep · · · replica(s) (including the master copy) per block.
3) We have · 1 · split(s) per machine. 1 block 1 split
4) Mapper0 reads · 2 · records.
   Mapper1 reads · 2 · records.
5) Memory content during each spill in $Machine0$:

Spill1	Spill2	Spill3	Spill4
[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]

Spills in $Machine1$:

Spill1	Spill2	Spill3	Spill4
[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]

6) a) Partitions in $Machine0$:

Spill1		Spill2		Spill3	
Partition0	Partition1	Partition0	Partition1	Partition0	Partition1
[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]

Partitions in $Machine1$:

Spill1		Spill2		Spill3	
Partition0	Partition1	Partition0	Partition1	Partition0	Partition1
[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]

b) Partitions in $Machine0$:

Spill1		Spill2		Spill3	
Partition0	Partition1	Partition0	Partition1	Partition0	Partition1
[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]	[,][,] [,][,]

Partitions in $Machine1$:

Spill1		Spill2		Spill3	
Partition0	Partition1	Partition0	Partition1	Partition0	Partition1
[,][,]	[,][,]	[,][,]	[,][,]	[,][,]	[,][,]
[,][,]	[,][,]	[,][,]	[,][,]	[,][,]	[,][,]

c) Files in $Machine0$:

File0.0	File0.1	File0.2
[,][,][,][,]	[,][,][,][,]	[,][,][,][,]
File0.3	File0.4	File0.5
[,][,][,][,]	[,][,][,][,]	[,][,][,][,]

Files in $Machine1$:

File1.0	File1.1	File1.2
[,][,][,][,]	[,][,][,][,]	[,][,][,][,]
File1.3	File1.4	File1.5
[,][,][,][,]	[,][,][,][,]	[,][,][,][,]

7) Merges in $Machine0$:

Merge0	Merge1
[,][,][,][,][,]	[,][,][,][,][,]

Merges in $Machine1$:

Merge0	Merge1
[,][,][,][,][,]	[,][,][,][,][,]

8) Files in $Machine0$:

File0.0	File0.1
[,][,][,][,][,]	[,][,][,][,][,]

Files in $Machine1$:

File1.0	File1.1
[,][,][,][,][,]	[,][,][,][,][,]

9) Reducer0 reads $\cdots$ from $Machine0$ and $\cdots$ from $Machine1$.
Reducer1 reads $\cdots$ from $Machine0$ and $\cdots$ from $Machine1$.

10) Merges in $Machine0$:

Merge0	Merge1
[,{ }][,{ }][,{ }][,{ }]	[,{ }][,{ }][,{ }][,{ }]

Merges in $Machine1$:

Merge0	Merge1
[,{ }][,{ }][,{ }][,{ }]	[,{ }][,{ }][,{ }][,{ }]

11) Reduce function is executed ··· times in $Machine0$.
Reduce function is executed ··· times in $Machine1$.

12) Files in $Machine0$:

Output0
[,] [,] [,] [,]

Files in $Machine1$:

Output1
[,] [,] [,] [,]

2. Let's suppose that we have a cluster of 100 machines and a MapReduce job with 1.000.000 key-value pairs in the input that generate 100.000 pairs in the output. Assume that both map and reduce functions generate one pair in the output per call. Assuming the reduce function is commutative and associative, is it worth to use the combine function? Briefly justify your answer.

not good
on average, each key will appear 10 times
if there are more workers, different values with same key being in same machine will be low

3. Let's consider that the CPU time to process a key-value pair is always zero (both in the map and the reduce functions), as well as the time to retrieve a disk block, and we can send control messages between machines without any cost. However, our network bandwidth is 10MB/sec and we only have one processor per machine. We have ten machines, which have a direct connection between them (i.e., sending data from machine A to machine B is always one hop away, for any pair of machines, as depicted in the figure).

We are launching a MapReduce job with a mapper and a reducer in each of these machines and an input file of key-value pairs of an overall size of 1GB[8] (whose chunks are already uniformly distributed in the ten machines, without any replication and not necessary of default size). The map function generates exactly one key-value pair per call, with exactly the same size as the key-value pair in the input. Assume that the hash function (without any cost) used in the partition phase perfectly maps the values in a uniform way; and also that when two machines are transferring data from one to another, they cannot perform any other data transfer activity in parallel (i.e., while A is transferring data to B, neither A nor B can send or receive anything from anyone else until this transfer is over, A can only send to B and not receive anything from anyone, B can only receive from A and not send anything to anyone, not even A). Supposing that scheduling the data transfer in the cluster of ten machines is perfect and there is not any idle period for any of the machines, which is the **duration of the job**? Explicit any **calculation and assumption** you need to make.

is ans 180 or 18?

[8]Assume 1GB=1000MB.

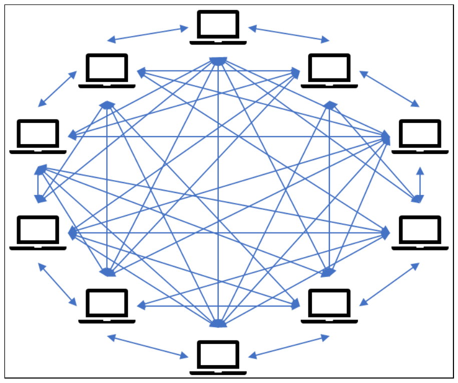


Figure 5: Fully connected architecture

Concept	**Cost**
Process any key-value pair	0s
Retrieve a disk block	0s
Send control messages between machines	0s
Network bandwidth	10MB/s

7.3 Spark in use

7.3.1 Problems

1. Consider a file (*wines.txt*) containing the following data:

   ```
   wines.txt
   type_1,2.064254
   type_3,2.925376
   type_2,2.683955
   type_1,2.991452
   type_2,2.861996
   type_1,2.727688
   ```

   Provide the ordered list of Spark operations (no need to follow the exact syntax, just the kind of operation and main parameters) you'd need to retrieve the minimum value per type. Do not use SQL and minimize the use of other Python libraries or code. Save the results in *output.txt*.

2. Consider two files containing the following data:

   ```
   Employees.txt
   EMP1;CARME;400000;MATARO;DPT1
   EMP2;EUGENIA;350000;TOLEDO;DPT2
   EMP3;JOSEP;250000;SITGES;DPT3
   EMP4;RICARDO;250000;MADRID;DPT4
   EMP5;EULALIA;150000;BARCELONA;DPT5
   EMP6;MIQUEL;125000;BADALONA;DPT5
   EMP7;MARIA;175000;MADRID;DPT6
   EMP8;ESTEBAN;150000;MADRID;DPT6
   ```

   ```
   Departments.txt
   DPT1;DIRECCIO;10;PAU CLARIS;BARCELONA
   DPT2;DIRECCIO;8;RIOS ROSAS;MADRID
   DPT3;MARKETING;1;PAU CLARIS;BARCELONA
   DPT4;MARKETING;3;RIOS ROSAS;MADRID
   DPT5;VENDES;1;MUNTANER;BARCELONA
   DPT6;VENDES;1;CASTELLANA;MADRID
   ```

   Provide the ordered list of Spark operations (no need to follow the exact syntax, just the kind of operation and main parameters) you'd need to retrieve for each employee his/her department information. Do not use SQL and minimize the use of other Python libraries or code. Save the results in *output.txt*.

3. Consider an error log file (*log.txt*) like the one bellow:

   ```
   log.txt
   20150323;0833;ERROR;Oracle
   20150323;0835;WARNING;MySQL
   20150323;0839;WARNING;MySQL
   20150323;0900;WARNING;Oracle
   20150323;0905;ERROR;MySQL
   20150323;1013;OK;Oracle
   20150323;1014;OK;MySQL
   20150323;1055;ERROR;Oracle
   ```

   Provide the ordered list of Spark operations (no need to follow the exact syntax, just the kind of operation and main parameters) you'd need to retrieve the lines corresponding to both *errors* and *warnings*, but adding *Important:* at the beginning of the identifier of those of errors (i.e., only *errors*). Do not use SQL and minimize the use of other Python libraries or code. Save the results in *output.txt*.

1. use the lit() function to create a column with a constant value
2. These are all transformations. For spark code to execute, action is needed like show() or something else

4. Assume that "spark" variable is a **Spark** session and the `dataset.csv` contains the two columns "Color" and "Radius". Clearly **identify the problems** you find in the following Spark code and propose some fix to obtain the expected result.

```
df0= spark.read.csv("dataset.csv", \
                    header='true', \
                    inferSchema='true', \
                    sep=',')
df1 = df0.select("Color", "Radius")
df2 = df1.withColumn("Pi", 3.141592)    lit(3.14)
df3 = df2.withColumn("Area", df2.Radius*df2.Radius*df2.Pi)
df4 = df3.groupBy("Color").max("Area")
df5 = df4.sort("Color")
```

5. Given two files containing the following kinds of data:

Employees.txt with fields: EmployeeID; EmployeeName; YearlySalary; CityOfResidence; SiteOfWork
EMP4;RICARDO;250000;MADRID;DPT4
EMP5;EULALIA;150000;BARCELONA;DPT5
EMP6;MIQUEL;125000;BADALONA;DPT5
EMP7;MARIA;175000;MADRID;DPT6
EMP8;ESTEBAN;150000;MADRID;DPT6
...

Departments.txt with fields: SiteID; DepartmentName; StreetNumber; StreetName; City
DPT1;DIRECCIO;10;PAU CLARIS;BARCELONA
DPT2;DIRECCIO;8;RIOS ROSAS;MADRID
DPT3;MARKETING;1;PAU CLARIS;BARCELONA
DPT4;MARKETING;3;RIOS ROSAS;MADRID
...

Consider the following **PySpark code** and answer the questions bellow.

```
source1 = spark.read.format("csv").load("employees.txt", header='false', inferSchema='true', sep=";")
source2 = spark.read.format("csv").load("departments.txt", header='false', inferSchema='true', sep=";")
A = source1.toDF("eID","eName","eSalary","eCity","eDpt")
B = source2.toDF("dID","dArea","dNumber","dStreet","dCity")
C = A.select(A.eCity.alias("city"))
D = B.select("dArea")
E = D.crossJoin(C)
F = B.select("dArea",B.dCity.alias("city"))
G = E.subtract(F)
H = G.select("dArea")
result = D.subtract(H)
```

Area names for those departments located in all cities where employees live (regardless if employees are from other departments)

(a) State in natural language the corresponding query it would answer?

D=B.select("dArea") will be reading file B.select("dArea",B.dCity.alias("city")) will again read file
to solve this, we need to do B.cache() -> for this we need to do B.cache() or B.persist()

(b) Clearly indicate any mistake or improvement you can fix/make in the code? For each of them give (1) the line number, (2) pseudo-code to implement the fix, and (3) brief rationale.

6. Given two files containing the following kinds of data: Did not understand question

Employees.txt with fields: EmployeeID; EmployeeName; YearlySalary; CityOfResidence; SiteOfWork
EMP1;RICARDO;250000;MADRID;DPT1
EMP2;EULALIA;150000;BARCELONA;DPT2
EMP3;MIQUEL;125000;BADALONA;DPT3
EMP4;MARIA;175000;MADRID;DPT4
EMP5;ESTEBAN;150000;MADRID;DPT3
...

Departments.txt with fields: SiteID; DepartmentName; StreetNumber; StreetName; City

    DPT1;DIRECCIO;10;PAU CLARIS;BARCELONA
    DPT2;DIRECCIO;8;RIOS ROSAS;MADRID
    DPT3;MARKETING;1;PAU CLARIS;BARCELONA
    DPT4;MARKETING;3;RIOS ROSAS;MADRID
    ...

Give a sequence of Spark operations in pseudo-code (resembling PySpark) to obtain for each city where employees that work in a site of a department in Barcelona live, the sum of the salaries of those employees. The result for the exemplary data would be:

MADRID;400000

BADALONA;125000

...

7. Consider three files containing the following kinds of data:

Employees.txt
EMP1,CARME,400000,MATARO,DEPT1,PROJ1
EMP2,EULALIA,150000,BARCELONA,DEPT2,PROJ1
EMP3,MIQUEL,125000,BADALONA,DEPT1,PROJ3

Projects.txt
PROJ1,IBDTEL,TV,1000000
PROJ2,IBDVID,VIDEO,500000
PROJ3,IBDTEF,TELEPHONE,200000
PROJ4,IBDCOM,COMMUNICATIONS,2000000

Departments.txt
DEPT1,MANAGEMENT,10,PAU CLARIS,BARCELONA
DEPT2,MANAGEMENT,8,RIOS ROSAS,MADRID
DEPT4,MARKETING,3,RIOS ROSAS,MADRID

Provide the ordered list of Spark operations (no need to follow the exact syntax, but just the kind of operation and main parameters) you would need to obtain the departments with all employees assigned to the same project. The result must include department number. Save the results in `output.txt`. In the previous example, the result should be *DEPT2* and *DEPT4*.

8. Consider two files containing the following kinds of data:

Employees.txt
EMP4;RICARDO;250000;MADRID;DPT4
EMP5;EULALIA;150000;BARCELONA;DPT5
EMP6;MIQUEL;125000;BADALONA;DPT5
EMP7;MARIA;175000;MADRID;DPT6
EMP8;ESTEBAN;150000;MADRID;DPT6

Departments.txt
DPT1;DIRECCIO;10;PAU CLARIS;BARCELONA
DPT2;DIRECCIO;8;RIOS ROSAS;MADRID
DPT3;MARKETING;1;PAU CLARIS;BARCELONA
DPT4;MARKETING;3;RIOS ROSAS;MADRID

Did not understand question

Provide the ordered list of Spark operations (no need to follow the exact syntax, but just the kind of operation and main parameters) you'd need to retrieve the list of department IDs for those departments with workers from all cities where there are employees. Save the results in `output.txt`.

9. Consider two files containing the following kinds of data:

Employees.txt
EMP1;RICARDO;250000€;MADRID;SITE2
EMP2;EULALIA;150000€;BARCELONA;SITE1
EMP3;MIQUEL;125000€;BADALONA;SITE3
EMP4;MARIA;175000€;MADRID;SITE2
EMP5;ESTEBAN;150000€;MADRID;SITE4

```
Departments.txt
SITE1;DPT.MANAGEMENT;FLOOR10;ST.PAU CLARIS;BARCELONA
SITE2;DPT.MANAGEMENT;FLOOR8;ST.RIOS ROSAS;MADRID
SITE3;DPT.MARKETING;FLOOR1;ST.PAU CLARIS;BARCELONA
SITE4;DPT.MARKETING;FLOOR1;ST.RIOS ROSAS;MADRID
SITE5;DPT.MARKETING;FLOOR5;ST.MARTI PUJOL;BADALONA
```

Provide the ordered list of Spark operations (no need to follow the exact syntax, but just the kind of operation and main parameters) you'd need to retrieve the list of department IDs for those departments with sites in all cities where employees live (these employees can be even from other departments). Save the results in `output.txt`. In the previous example, the result should be *DPT.MANAGEMENT*, because it has sites in MARKETING all the three cities where there are employees (i.e., MADRID, BARCELONA and BADALONA). However, *DPT.MANAGEMENT* should not be in the result, because it does not have any site in BADALONA, where EMP3 lives.

10. Consider three files relating to a bibliographic database: `author.csv` relates authors with papers (you may assume that author names are unique, that authors have one or more papers, and that papers have one or more authors); `title.csv` gives the title of a paper (you may assume a paper has one title, but one title may be shared by many papers); and `citation.csv` indicates which papers cite which other papers (you may assume that each paper cites at least one other paper, that a paper may be cited zero or more times, and that a paper cannot cite itself).

`author.csv`

AUTHOR	PAPERID
...	...
C. Gutierrez	GP2014
C. Gutierrez	AGP2013
C. Gutierrez	GZ2011
...	...
J. Perez	GP2014
J. Perez	AGP2013
J. Perez	P2017
...	...
R. Angles	AGP2013
R. Angles	AKK2016
...	...

`title.csv`

PAPERID	TITLE
...	...
GP2014	Semantics of SPARQL
AGP2013	Deduction for RDF
GZ2011	Graph databases
...	...

`citation.csv`

CITES	CITED
...	...
GP2014	AGP2013
AGP2013	GZ2011
P2017	AKK2016
...	...

The headers are shown for illustration here. They do not need to be considered.

The count of self-citations for an author A, denoted $\mathsf{self}(A)$, is defined as the number of citation pairs (P_1, P_2) where A is an author of both. The count of citations given by an author A, denoted $\mathsf{give}(A)$, is the count of citation pairs (P_1, P_2) such that A is an author of P_1. The count of citations received by A, denoted $\mathsf{receive}(A)$, is the count of citation pairs (P_1, P_2) where A is an author of P_2. The ratio of self-citations to all citations given and received are then defined, respectively, as $\frac{\mathsf{self}(A)}{\mathsf{give}(A)}$ and $\frac{\mathsf{self}(A)}{\mathsf{receive}(A)}$. In case that $\mathsf{receive}(A) = 0$, you should omit the author A from the results (note that $\mathsf{give}(A)$ cannot be 0 as an author must have at least one paper and a paper must have at least one citation). We provide an example output for the input data:

AUTHOR	SELFGIVERATIO	SELFRECEIVERATIO
C. Gutierrez	1.000	1.000
J. Perez	0.333	1.000
R. Angles	0.000	0.000
...	...	

Headers are only shown for illustration. We will use Apache Spark to perform the analysis and compute the output. You should not assume any ordering of the input files. You do not need to order the output file in any particular way.

Given this input and desired output, design a Spark process to complete the required processing. In particular, you should draw the high-level DAG of operations that the Spark process will perform, detailing the sequence of transformations and actions. You should briefly describe what each step does, clearly indicating which steps are transformations and which are actions. You should also indicate which RDDs are virtual and which will be materialized. You should use caching if appropriate. You should provide details on any functions passed as arguments to the transformations/actions you use.

7.4 Spark internals

7.4.1 Theoretical questions

1. What indicates to Spark query optimizer the end of a stage and the beginning of the next one?

   The end of a Spark stage is indicated when a wide dependency is encountered

7.4.2 Problems

In notion

1. Considering the file and result example, briefly indicate the problems you find in the Spark code below (if any), and **modify the code** to fix them (if needed).

1. Average is not being computed properly
2. Action not performed so code will not be executed

```
Employees.txt
EMP1;CARME;40000;MATARO;DPT1;PROJ1
EMP2;EUGENIA;35000;TOLEDO;DPT2;PROJ1
EMP3;JOSEP;25000;SITGES;DPT3;PROJ2
EMP4;RICARDO;25000;MADRID;DPT4;PROJ2
EMP5;EULALIA;15000;BARCELONA;DPT5;PROJ2
EMP6;MIQUEL;12500;BADALONA;DPT5;PROJ3
EMP7;MARIA;17500;MADRID;DPT6;PROJ3
EMP8;ESTEBAN;15000;MADRID;DPT6;PROJ3
```

```
Expected result
[ PROJ1,[ EUGENIA, 35000, 37500.0 ] ]
[ PROJ2,[ EULALIA, 15000, 21666.6 ] ]
[ PROJ3,[ MIQUEL, 12500, 15000.0 ] ]
```

```
rawEmps = sc.textFile(r"Employees.txt")
emps = rawEmps.map(lambda l: tuple(l.split(";"))).cache()
RDD1 = emps.map(lambda t: (t[5], (int(t[2]), 1)))
RDD2 = RDD1.reduceByKey(lambda t1, t2: (t1[0] + t2[0])/(t1[1] + t2[1]))
RDD3 = emps.map(lambda t: (t[5], t))
RDD4 = RDD3.join(RDD2)
RDD5 = RDD4.filter(lambda t: int(t[1][0][2])<t[1][1])
RDD6 = RDD5.map(lambda t: (t[0],(t[1][0][1],t[1][0][2],t[1][1])))
Result = RDD6.sortByKey()
```

2. Consider the following pipeline. This pipeline runs in a 4-machine cluster with HDFS to store the files and Spark to execute it. `File1` and `File2` are distributed in the cluster in 6 chunks each.

```
RddF1:= sc.textFile("...file1.txt")
RddF2:= sc.textFile("...file2.txt")
Rdd2:= RddF1.mapToPair(s.split(";")[0], s.split(";")[1-2])
Rdd3:= Rdd2.GroupByKey()
Rdd4:= Rdd3.MapValues(f1)
Rdd5:= RddF2.mapToPair(s.split(";")[0], s.split(";")[1-2])
Rdd6:= Rdd5.GroupByKey()
Rdd7:= Rdd6.MapValues(f2)
Rdd8:= Rdd4.join(Rdd7)
Rdd8.save("...file3.txt")
```

not clear
what if map??

   a) How many stages will the scheduler generate for this pipeline? (Justify your answer) 3 stages

   b) How many tasks will be generated within each stage? (Justify your answer)

Tasks are divided per chunk
Since, there are 6 chunks, 6 tasks will be generated

3. Consider the legacy code written in MapReduce that specifies a `map()`, `combine()` and `reduce()` operations. This job reads from a text file f_1. The combine and reduce operations coincide and you can assume all functions are correct. Write a Spark pipeline **equivalent to the MapReduce** job. Use `fmap` and `freduce` to refer to the code executed inside the map and combine / reduce operations. You can parametrise the `fmap` and `freduce` functions but resulting in minimal code adaptation.

4. Given a file of $3.2GB$ stored in an HDFS cluster of 50 machines, and containing $16 \cdot 10^5$ key-value pairs in a SequenceFile; estimate the execution time of a Spark job containing a single map transformation and an action storing the results in a file. Explicit any assumption you make and consider also the following parameters:

   64 sec

   - Chunk size: $128MB$ (default)
   - Replication factor: 3 (default)
   - Map function (i.e., the parameter of the transformation) execution time: 10^{-3}sec/call (this is **the only cost** you have to consider)
   - Save action execution time: 0sec (do not consider its cost at all)

8 Streams

8.1 Problems

1. Let's suppose an arrival rate $\lambda = 50 messages/sec$ and a service rate $\mu = 50 messages/sec$. Given the message arrivals in the table bellow and a size of $100 bytes$ per message, which is the minimum buffer size needed to guarantee that no message is lost?

Instant (sec)	1	2	3	4	5	6	7	8	9	10
#Messages	2	10	100	50	20	75	150	100	40	10

2. Let's suppose a stationary situation in the processing of a data stream, where the whole memory available is occupied by messages. Assume also the existence of a separated buffer big enough to allow us the complete processing of these messages already in memory without being concerned with new arrivals. Assume we have $M = 10$ memory pages with $R_S = 10$ messages per page. If the processing of these messages consists only of a lookup and in-memory processing/comparison of a message and a tuple, whose execution time is thousand times smaller than a disk access, give the cost in these two situations considering that every disk block of the table contains also $R_T = 10$ tuples:

   a) Going through an index for each message with cost $h = 3$ for the index and one more disk access to the lookup table per message.

   b) Bringing each block of the lookup table (whose size is $B = 100$) into memory one by one and checking all messages in memory against all its tuples.

3. Let's suppose that setting the execution environment for the processing of messages in a stream is $S = 100ms$ (e.g., placing&retrieving all the information to/from the stack), independently of their number. Both packing and unpacking the messages in a batch (i.e., a list of elements) have the same cost of $Pk = 1ms$ per message, and processing each message is $Ps = 10ms$.

   a) Which is the cost of processing truly streaming 10 messages one at a time?

   b) Which is the cost of processing one packed micro-batch of 10 messages at once?

4. Let's suppose we have a log file recording the events coming from different machines. Thus, for each event we have the following information:

$$(logID, traceID, eventID, duration)$$

The $logID$ corresponds to the IP of the machine; the $traceID$ identifies the transaction inside the machine (i.e., two traceIDs can coincide in different machines); the $eventID$ identifies the kind of action performed by the machine; finally, the $duration$ is the number of milliseconds taken to implement the action. Assuming that we cannot keep all log entries in memory, and we decide to randomly sample them, **give the attributes** (up to three) you would use as parameters of the hash function implementing such sampling, so that each of the following queries gives a result as accurate as possible.

   a) For each machine, estimate the average number of actions per transaction. logId, traceId

   ...

   ...

   ...

b) Estimate the average sum of the duration per transaction. logId, traceId

. . .
. . .
. . .

c) Estimate the fraction of transactions where the same kind of action appears more than once.

. . . logId, traceId, eventId
. . .
. . .

d) Estimate the number of transactions with more than two actions taking more than 100ms.

. . .
. . . logId, traceId, duration
. . .

5. Let's suppose we have a black-list of IP addresses (whose packages we do not want to cross our firewall), which is too long to be kept in memory (10^7 elements). Thus, we decide to implement a **Bloom filter** (to avoid further processing of black-listed addresses), with only 10^8 bits.[9]

a) How many hash functions would you use? 7

b) What's the probability of a false positive in that case?

c) Briefly explain what is the consequence of a false positive in this concrete case.
Elements which do not need to be blocked might also be blocked
important information might be lost

6. Let's suppose we have a log file recording the events coming from different machines. Thus, for each event we have the following information:

$$(logID, traceID, eventID, duration)$$

The $logID$ corresponds to the IP of the machine; the $traceID$ identifies the transaction inside the machine; the $eventID$ identifies the kind of action performed by the machine; finally, the $duration$ is the number of milliseconds taken to implement the action. Consider the following table and assume that each machine generates the same number of events and at the same pace, and use an **exponentially decaying window model** with a constant $c = 0.5$.

Time	logID	traceID	eventID	duration
1	1	1	A	1
2	2	1	B	100
3	1	1	C	10
4	2	1	D	10
5	1	1	A	100
6	2	1	C	1

a) Give the milliseconds (and details of calculation) used per machine (i.e., for Machine 1 and Machine 2) when the last event arrives, provided that the current milliseconds are more valuable than the oldest ones.

[9]Note: $\ln 2 \approx 0.693$

7. Assume we ingest a stream with an event every time a ticket is sold at a theater. Precisely, the stream has the following structure:

$$(movieID, theaterID, timestamp, price)$$

Next, we ingest the following ordered set of events:

- $(m1, t1, 12h, 10€)$
- $(m2, t1, 14h, 12€)$
- $(m2, t2, 15h, 18€)$
- $(m1, t3, 18h, 6€)$
- $(m3, t2, 19:15h, 13€)$
- $(m1, t3, 19:30h, 10€)$
- $(m2, t3, 19:45h, 25€)$
- $(m3, t1, 20h, 17€)$
- $(m1, t3, 20:30h, 10€)$
- $(m2, t2, 21h, 8€)$

Provide a detailed answer (i.e., describe the process) for the following questions:

(a) Which theaters would be considered heavy hitters by the algorithm, given a required frequency of 50%?
t3

(b) Under the exponentially-decaying window model, using an aging constant of $c = 0.1$ and a purging threshold of 0.6 (i.e., we'll remove movies as soon as their popularity falls strictly bellow 0.6). What would be the most popular movies at 21h, considering an aging tick every 15 minutes? Give the value corresponding for each of the movies in that moment.
one counter for each keys
counter as much as movies

8. If you have a stream of messages that arrive at a rate of 10,000 per second accounting for $512MB$ per hour, how many RDDs will process your Spark Streaming process in one hour? Explicit all the assumptions you make.

9. Assume we ingest a stream with an event every time a ticket is sold at a theater. Precisely, the stream has the structure $(movieID, theaterID, timestamp, price)$. Next, we ingest the following ordered set of events:

- Event 1: $(m_3, t_4, 12h, 10\$)$
- Event 2: $(m_1, t_2, 13h, 17\$)$
- Event 3: $(m_2, t_4, 14h, 11\$)$
- Event 4: $(m_4, t_1, 15h, 8\$)$
- Event 5: $(m_1, t_3, 16h, 9\$)$
- Event 6: $(m_3, t_4, 17h, 5\$)$
- Event 7: $(m_6, t_1, 18h, 15\$)$
- Event 8: $(m_5, t_2, 19h, 12\$)$
- Event 9: $(m_7, t_5, 20h, 17\$)$
- Event 10:$(m_1, t_1, 21h, 11\$)$

Which theaters would be considered heavy hitters (using the approximate method) considering a required frequency of 33%? Provide a detailed answer (i.e., describe the content of the auxiliary structure after processing every message).

We do not need in advance what are the analysis that we need to perform to solve that problem

9 Architectures

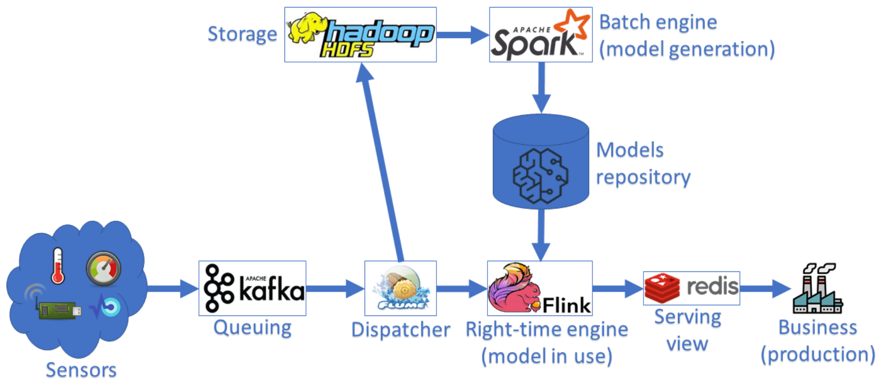
9.1 Theoretical questions

Solve problems like
high entry barrier
permanent transformation
fixed schema

1. What problem do Data Lakes solve?

9.2 Problems

1. Consider the software architecture below and briefly explain the main **management (maintenance) risk** it has.



1. Inconsistency
2. Transformation applied by spark and flink may be different
3. Implementation in beginning can be same. But in long term, it is difficult to maintain consistency in 2 different places.

2. Assume we have a **MongoDB** collection in a distributed cluster, which contains prices of apartments without any secondary index. Such collection is big enough not to completely fit in memory. We want to **use Spark** to compute the standard deviation per neighbourhood. Clearly identify the most efficient option and briefly justify the choice (it is **not necessary to provide the Spark code**).

   (a) Use Spark only to push the query to MongoDB aggregation framework and simply get the result.

   (b) Push only some of the operations to MongoDB aggregation framework and the run the rest in Spark.

   (c) Load the whole collection to an RDD and perform all computations in Spark.

A. No, because we would need the mongo router to perform all the work, losing the power of parallelism. MongoDB parallelizes something but not everything.

B. Yes. Push down selection and projection as much as possible in MongoDB and send only needed data in Spark

C. No, because we would need to transfer everything to Spark, implying a huge data transfer overload.