大数据数据库

## 解释 NoSQL 数据存储出现的主要原因。

1. Larger datasets.
2. Unstructured and semi-structured datasets more common.
3. Frequent schema changed overtime.
4. Better distribution and continuously available.

## 列出并描述 NoSQL 数据存储的主要特征。

1. Flexible non-relational data modes.
2. High read and write scalability.
3. Easy to distribute, horizontal scaling.
4. High availability.

## 解释 ACID 和 BASE 属性之间的区别。

According to the **CAP theory**: **it is impossible to achieve both consistency and availability** in a partition tolerant **distributed system.**

1. **ACID**: provides a consistent system

*Atomicity*: A transaction is either **executed completely or not at all**.

*Consistency*: A database that is in **consistent** state **before** a transaction and **after** a transaction.

*Isolation*: A transaction **executed in isolation** from the other transactions.

*Durability*: **Changes** in the database made by a committed **transaction** are **permanent**.

1. **BASE**: provides high availability.

*Basically Available:* An application works basically all the time.

*Soft state:* Due to the lack of immediate consistency, **data values may change over time.**

*Eventual consistency:* the system will become consistent over time, given that the system doesn't receive input during that time.

‘BASE does not enforce immediate consistency does not mean that it never achieves it’

**<https://phoenixnap.com/kb/acid-vs-base>**

## 讨论分布式数据存储设置中一致性和可用性之间的权衡。

For consistency, all DB instances will always contain the same data, and all writes must duplicate before writing operation is complete, all databases are "locked" until the write is done. Consistency will make availability slower.

## 讨论不同的一致性模型以及为什么需要它们。

Consistent hashing

Keys (data) and nodes (servers) map to the same ID-space.

Create a metric space (a ring). Then hash all the servers to fit in the ID-space. Then hash all the keys (data) into the same space. When a request for a key (data) comes in, it simply choses the first node (server) in clock-wise order after the key in the ID-space.

One can also add a replication factor r, meaning that the next r nodes are responsible for the key (data).

It's needed because we want to:

- Minimize the number of nodes to be copied after a configuration change.

- Incorporate hardware characteristics into hashing model

## 解释 CAP 定理。

The CAP theorem: it is impossible for a distributed system to simultaneously **provide more than two out of the following three guarantees**:

1. **Consistency**: After update, all readers in a distributed system see the same data. All nodes always contain the same data.
2. **Availability**: **All requests will be answered**, regardless of **crashes** or **downtime**.
3. **Partition tolerance**: It is the ability of a data processing system to **continue processing data** even if a **network partition** causes **communication errors** between subsystems.

## 解释垂直和水平可扩展性之间的区别。

Horizontal scaling:

a. **Increase the number of computers**/nodes

b. it is more **popular** because of **low-cost hardware.**

Vertical scaling:

a. **Add resources to a single node (CPU, RAM etc.)**

b. Increase number of threads to solve problems.

## 根据数据模型，列出并描述 NoSQL 数据存储的主要特点和应用。

a. Key-Value

- Schema-free

---- Keys are unique

---- Values of arbitrary types

- Efficient in storing distributed data

- Limited query facilities and indexing

get(key) and put (key, value)

- Used for

---- Web session information

---- User profiles

---- Shopping cart data

b. Column-family

- Schema-free

---- Rows have unique keys

---- Each row points to one/many column families

---- Each column familj consists of column(s)

---- Each column has a value

- Better than key-value stores for querying and indexing

- Used for

---- Document stores applications

---- Analytics scenarios (e.g., web analytics)

c. Document

- Schema-free

---- Keys are unique

---- Values are documents (JSON etc.)

- Indexing and querying based on primary keys and content

- Used for:

---- Items with similar nature, different structure

---- Blogging platforms

---- CMS

---- Fast application development

d. Graph databases

- Graph model

---- Nodes and edges

---- Consists of key-value pairs

-Suitable for very interconnected data since they are efficient in traversing relationships

- Not as efficient as other NoSQL databases

- Used for:

---- Location-based services

---- Recommendation engines

---- Complex network-based applications

并行计算

## PAR-Q1：定义以下技术术语：（彻底和笼统。示例不是定义。）

集群（在高性能和大数据计算中）

A cluster is **a group of computers** that are **independent** of each other, **interconnected by a high-speed network,** that form a group and are managed in a single system model. When a client interacts with a cluster, the cluster acts like a standalone server. Cluster configurations are used to improve **availability** and **scalability**.

并行工作（并行算法）

A parallel algorithm is a method and procedure for **solving a problem jointly using multiple processors**. It is performed by first **decomposing a given problem into a number of sub-problems** that are **as independent as possible**, and then **using multiple computers to solve it at the same time** in order to **obtain a solution to the original problem.**

并行加速

Text

Description automatically generated

Speedup achieved by a parallel algorithm is defined as the **ratio of the time required by the best sequential algorithm** to solve a problem, T(1), **to the time required by parallel algorithm** using **p processors** to solve the same problem,T(p). For simplicity, T(1) is calculated by running the parallel program on one processor.

通信延迟（用于从节点 Pi 向节点 Pj 发送消息）

网络中的延迟含义最好被认为是一个数据包被捕获、传输、通过多个设备处理，然后在其目的地接收和解码所需的时间量。

the amount of time it takes for **a packet of data** to be **captured**, **transmitted**, **processed** through multiple devices, then **received** at its destination and **decoded**.

时间数据局部性

在相对较短的时间内重复使用特定数据和/或资源。

**Reuse of specific data** and/or resources **in a relatively short period of time.**

动态任务调度

任务的分派和可调度性测试都是在系统运行时在线进行的。

动态调度与静态调度相比有更好的灵活性，然后由于可调度性测试需要在线进行，它的调度算法的复杂度不能太高，并且由 于无法保证是否可以被调度，算法的可预测性(predictability)很差。

**Task assignment and schedulability tests are performed online while the system is running.**

Dynamic scheduling has better **flexibility** than static scheduling, but the **complexity** of the scheduling algorithm cannot be too high because the schedulability test needs to be performed online, and the **predictability** of the algorithm is **poor** because there is no guarantee that it can be scheduled.

静态任务调度

静态调度方法，任务的分配离线进行，即在实时任务正式在处理机上调度执行前，先把任务在处理机上的分配和调度时间安排好,在任务正式开始执行后按照预先的调度方案执行。

预先安排好调动，减少任务调度过程中的开销;而缺点在于缺乏灵活性，在实际的调度中不能够及时地根据系统资源和任务的执行情况进行及时的调整。

In the static scheduling method, task assignment is performed offline, i.e., before the real-time task is formally scheduled on the processor, the assignment and scheduling time of the task on the processor are arranged first, and the task is executed according to the pre-scheduling scheme after the task formally starts execution.

The prearranged mobilization reduces the overhead in the task scheduling process; and the disadvantage is the lack of flexibility, which does not allow timely adjustment in the actual scheduling according to the system resources and the execution of tasks.

## PAR-Q2：解释以下并行算法范例：并行分治法。

This divides the **input set** of the problem into **subsets**, computes **partial** **results** for each subset, and finally uses the **merge** function to obtain the **final answer.**

这将问题的输入集划分为子集，计算每个子集的部分结果，最后使用合并函数获得最终答案。

## PAR-Q3：讨论在流式传输中使用大数据包大小和小数据包大小的性能影响。

Large packet sizes:

- Less overhead少开销

- If failure, more data must be resent

- Takes longer to receive, not suitable for time dependent streams

Small packet sizes:

- More overhead大开销

- If failure, less data must be resent

- Higher chance of failure as more packages are sent'

- Quicker to receive

## PAR-Q4：为什么数据中心中运行 I/O 密集型任务（例如文件/数据库访问）的服务器（集群节点）要运行的任务多于内核数量？

IO-intensive, tasks that involve network and disk IO are IO-intensive tasks. **These tasks consume very little CPU and most of the task's time is spent waiting for IO operations to complete** (because the speed of IO is much lower than the speed of CPU and memory). **For IO-intensive tasks, the more tasks there are, the more efficient the CPU is**, but there is a limit.

IO密集型，涉及到网络、磁盘IO的任务都是IO密集型任务，这类任务对CPU的消耗很少，任务的大部分时间都在等待IO操作完成（因为IO的速度远远低于CPU和内存的速度）。对于IO密集型任务，任务越多，CPU效率越高，但也有一个限度

简而言之，IO密集任务越多，CPU的效率越高，不能让CPU闲下来。

## PAR-Q5：在骨架编程中，您需要使用哪个骨架来计算大数组中的最大元素？绘制生成的伪代码（解释你的代码）。

## PAR-Q6：描述使用算法骨架的高级并行编程的优点/优势和缺点/限制。

Advantages:

- Abstraction, hiding complexity

- Parallelization for free

- Easier to analyze and transform

Disadvantages:

- Requires complete understanding and rewriting of a computation

- Available skeleton set does not always fit

- May lose some efficiency compared to manual parallelization

## PAR-Q7：推导出阿姆达尔定律并给出解释。

阿姆达尔定律表明并行算法的加速实际上受到必须顺序执行的操作数量的限制。

它计算的是相对加速比：

T(1): 处理器数量无关，最佳顺序算法解决问题所需的时间，通常为1个处理器运行并行算法处理任务的时间

T(p)：p个处理器运行并行程序处理任务的时间

并行算法实现的加速比

S(p) = T(1)/T(p)

再定义：

S：一个任务串行执行的部分

P：一个任务可并行处理的部分

N：处理器数量

T(1) = S + P

T(p) = S + P/N

加速比：S(p) = T(1) / T(p) = (S+P) / (S + P/N)

定义序列分数F = S/T(1)

得到PPT中的古斯塔夫公式，也就是重新表示并行算法实现的加速比：

S(p) = 1 / (F + (1-F)/N)

举例：假设我们有一个包含 100 个操作的程序，每个操作需要 1 个时间单位。如果可以并行完成 80 次操作，即 P = 80。20 次操作必须按顺序完成，即 S = 20。那么使用 80 个处理器，加速比将是 100 / 21 < 5

S(p) = (S + P) / (S + P/N) = (20 + 80) / (20 + 80/80) = 100/21 < 5

## PAR-Q8：相对和绝对并行加速有什么区别？其中哪个预计会更高？

Graphical user interface, text, application, email

Description automatically generated

预计相对加速会高，Tbest理论上接近无限小。

## PAR-Q9：PRAM（并行随机存取机）计算模型具有最简单的并行成本模型。它代表了现实世界并行计算机的哪些方面，又从哪些方面抽象出来？

Abstract:

- Scheduling overhead 调度开销

- Communication latency 通信时延

- Bandwidth limitations带宽限制

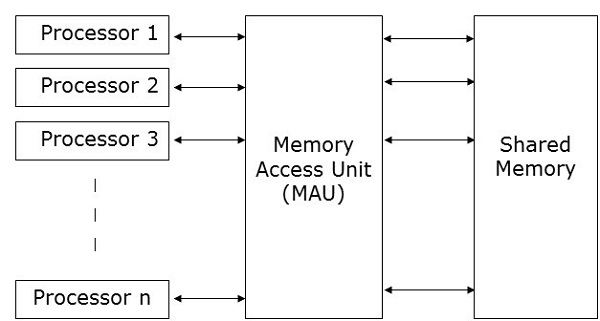
- Memory latency内存时延

代表了：

多个处理器

多个处理器连接到单个存储器块。PRAM模型包含：

1. 一组相似类型的处理器；
2. 所有处理器共享一个共同的内存单元。处理器只能通过共享内存进行通信。
3. 存储器访问单元（MAU）将处理器与单个共享存储器连接。



## PAR-Q10：流计算的哪个特性使得计算与数据传输重叠成为可能？

Streaming calculations: events should be processed as soon as they occur, not cached for batch processing.

High performance, real time, and low latency make it possible to overlap computation with data transfer.

流式计算：事件一发生就应该被处理，而不是被缓存起来进行批量处理。

高性能、实时性和低延迟使得计算与数据传输重叠成为可能。

