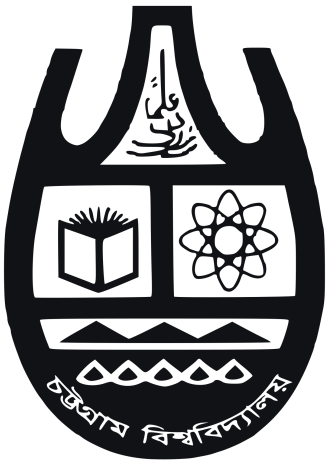
University of Chittagong



Dept. of Computer Science & Engineering

Course: CSE-813

Assignment on: Distributed and Cloud Computing

Submitted to

Professor Dr. Md. Hanif Seddiqui

Dept. of Computer Science & Engineering

University of Chittagong

Submitted By

Md. Zahin Uddin

ID: 17701052

Session: 2016-2017

Date: 29/03/2022

Ans.to.the.qus.no: 1

(a)

Qus: Define distributed and cloud systems? Give appropriate examples.

Ans:

A distributed computing system is a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network. Each processor and its resources are called a node or site or machine of the distributed system.

An electronic banking system, for example, is a distributed system in which nodes or machines are linked by an interconnected system and each node can send a message to the other.

A cloud system is a concept for providing on-demand network access to a shared configurable computing resource such as network servers, storage, applications, and services that can be swiftly supplied and released with minimal administrative effort or service provider contact.

Dropbox, for example, is a cloud-based solution that allows millions of users to store and save data without taking up more storage space on their personal computers or mobile devices.

(b)

Qus: Differentiate tightly and loosely coupled systems with appropriate Figures.

Ans:

Tightly Coupled System vs Loosely Coupled System:

1. The main difference between tightly coupled and loosely coupled systems is that tightly coupled systems have shared memory, whereas loosely coupled systems have distributed memory.

2. When processes operating on various processors have limited interaction, loosely connected is efficient.The tightly linked system, on the other hand, can handle a greater degree of interaction between processes and is more efficient for high-speed and real-time processing.
3. A tightly coupled system employs parallel computing, whereas a loosely coupled system employs distributed computing.
4. When compared to tightly coupled systems, the processors of distributed computing systems can be located at a variable distance from one another to cover a large geographical area.
5. Tightly coupled systems have one shared memory and the number of processors is limited by the bandwidth of the shared memory, whereas in a distributed system, the memory is expandable as needed, allowing for the addition of an almost unlimited number of processors.

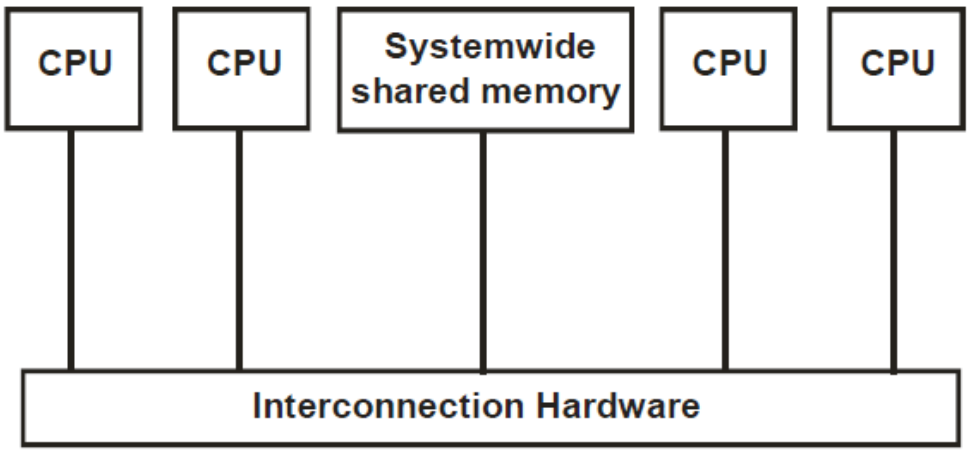


Figure: Tightly Coupled System

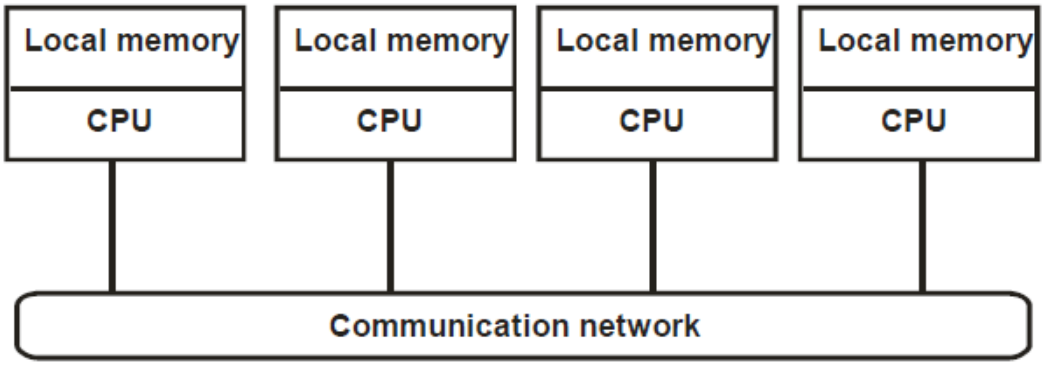


Figure: Loosely Coupled System

(c)

Qus: In what respect are distributed computing systems better than parallel processing systems?

Ans:

Distributed computing system is better than a parallel processing system in a number of ways.

1. Cost-effective

Despite their high implementation costs, distributed systems are cost-effective in the long run. In contrast to a mainframe computer, which has a single system consisting of several processors, a distributed system consists of several computers cooperating. This infrastructure is significantly less costly than a mainframe system.

2. Dependability

Distributed systems are far more reliable than single systems in terms of failures. Even if a single node fails, the remaining servers are unaffected. The other nodes can continue to function normally.

3. Efficiency

Distributed systems are meant to be efficient in every manner since they contain several computers. Each of these computers is capable of solving issues independently. This is not only efficient, but it also saves the user a considerable amount of time.

4. Extensibility

Distributed systems are built with scalability in mind. Users can add more workstations as their workload grows. There is no need to upgrade any of the systems. Furthermore, there are no limitations on the number of machines that can be used. This means that these machines will be capable of handling high-demand workloads with ease.

5. Latency

Distributed systems achieve low latency. The distributed system ensures that traffic from a node closer to the user is received by the system. As a result, the user may realize that serving them takes much less time.

Ans.to.the.qus.to: 2

(a)

Qus: How are location, relocation, and migration transparencies different from each other in distributed computing? Explain with examples.

Ans:

Location Transparency: This transparency conceals information about the physical location of the resource.

For example, the logical name google.com doesn't reveal the physical address of Google's server.

Migration Transparency: Migration transparency is said to be provided by distributed systems in which resources can be moved without affecting how the resources can be accessed. It conceals the fact that the resource may be moved from one location to another.

For instance, a resource may be moved from one machine to another while a user is accessing the resource.

Relocation Transparency: This transparency addresses the fact that resources can be relocated while the application is being used without the user being aware of it.

For example, a user might be using Wi-Fi on his mobile phone while moving from one place to another without getting disconnected.

(b)

Qus: Illustrate the commonly used techniques for implementing fault detection and recovery methods in a distributed operating system.

The fault detection and recovery strategy for increasing reliability entails employing hardware and software approaches to detect the onset of a problem and then restoring the system to a state suitable for continued operation.

The following are some of the most commonly used techniques for implementing this method in a distributed operating system:

1. **Atomic Transactions:** An atomic transaction is a computation consisting of a collection of operations that takes place indivisibly in the presence of failures and concurrent computations. That is, either all of the operations are performed successfully or none of their effects prevails. Transactions make crash recovery much easier because transactions can only end in two states. Either all the operations of the transaction are performed or none of the operations of the transaction is performed. In a system with a transaction facility, if a process halts unexpectedly due to a hardware error before a transaction is completed, the system subsequently restores any data objects that were undergoing modification to their original states. Without a transaction facility, it may be difficult or even impossible in some cases to roll back (recover) the data objects from their current inconsistent states to their original states.
2. **Stateless servers:** In distributed systems, the client-server architecture is widely employed to handle user requests. In this approach, a server may be developed utilizing either the stateful or stateless service paradigms. The two paradigms differ in one element of the client-server relationship: whether the history of served requests between a client and a server influences the execution of the next service request. The stateful technique is dependent on the history of served requests, whereas the stateless approach is not. In the case of a failure, stateless servers offer a unique advantage over stateful servers. That is because no client state information is kept, the stateless service architecture makes crash recovery incredibly simple.
3. **Acknowledgments and timeout-based retransmission of messages:** In a distributed system, events such as a node crash or a communication link failure may interrupt a communication that was in progress between two processes, resulting in the loss of a message. Therefore, a reliable interprocess communication mechanism must have ways to detect lost messages so that they can be retransmitted. Handling of lost messages usually involves the return of acknowledgment messages and retransmissions on the basis of timeouts. That is, the receiver must return an acknowledgment message for every message received, and if the sender does not receive any acknowledgment for a message within a fixed timeout period, it assumes that the message was lost and retransmits the message.

(c)

Qus: What are the design principles considered useful for better performance in a distributed system?

Ans:

A distributed system should perform better than or as well as the centralized system. For this purpose, the distributed system must be built efficiently.Some of the design principles considered useful for better performance are as follows:

1. Batch if possible: Batching frequently improves performance significantly. Transferring data across the network in huge chunks rather than individual pages, for example, is significantly more efficient. Similarly, throughout a sequence of messages transmitted between two communicating entities, piggybacking on the acknowledgment of prior messages with the following message enhances efficiency.

2. Cache whenever possible: Data caching at customers' locations typically enhances overall system performance because it makes data available wherever it is actively needed, saving a significant amount of computer time and network bandwidth.Furthermore, caching minimizes competition for centralized resources.

3. Minimize copying of data: Many procedures incur a significant CPU cost due to data copying overhead. Message data, for example, may travel the following path from its source to its recipient while being transferred:

1) From sender’s stack to its message buffer
2) From the message buffer in the sender’s address space to the message buffer in the kernel’s address space
3) Finally, from the kernel to the network interface board

On the receiving end, the data most likely follows a similar journey in the other direction.As a result, the message transfer process in this scenario involves a total of six copy operations. Similarly, the data copying cost for reading and writing operations on block I/O devices is high in various systems. As a result, it is preferable to avoid copying data for improved efficiency, albeit this is not always easy to do.

Making optimal use of memory management often helps in eliminating much data movement between the kernel, block I/O devices, clients, and servers.

4. Minimize network traffic: Reduced internode communication costs can also increase system performance. Access to remote resources, for example, necessitates communication, sometimes via intermediate nodes. As a result, relocating a process closer to the resources it uses the most may assist reduce network traffic in the system if the lower cost of accessing its preferred resources outweighs the potential rise in cost of reaching its less liked ones. Another method for reducing network traffic is to utilize the process migration feature to cluster two or more processes that often communicate with one another on the same system node. Avoiding the collecting of global status information for some choices also aids in the reduction of network traffic.

5. Take advantage of fine-grain parallelism for multiprocessing: Multiprocessing performance can also be enhanced by utilizing fine-grain parallelism. Threads, for example, are frequently used to organize server operations. Because they may simultaneously handle requests from several customers, servers built as a set of threads can work effectively. Another example of how this idea may be used to improve performance is fine-grained concurrency management of several processes using a common resource at the same time.

Ans.to.qus.no: 3

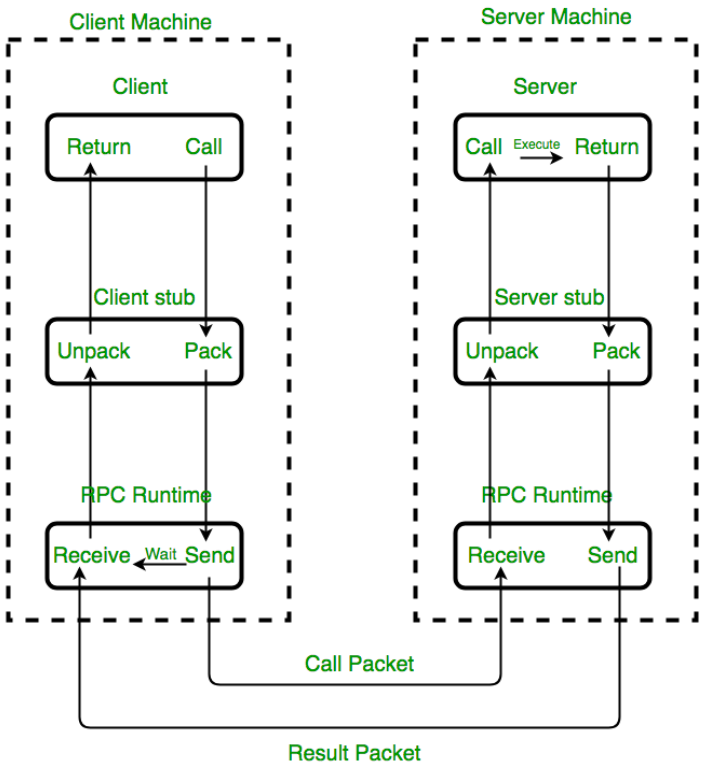
(a)

Qus: What is the primary motivation for the development of RPC? Draw a schematic diagram of an implementation of the RPC mechanism and illustrate the operation in brief.

Ans:

The main objective was to make distributed computation simple and efficient while still ensuring safe communication. Remote Procedure Call (RPC) is a protocol that allows one program to request a service from another program on a network without needing to know the network's specifics. RPC is a protocol that allows you to call other processes on distant systems from your local system.

Implementation of RPC mechanism: RPC is particularly well suited to client-server (e.g., query-response) interaction in which the flow of control alternates between the caller and the callee.



Implementation of RPC mechanism

The following steps take place during an RPC:

1. A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client's own address space.
2. The client stub marshalls(packs) the parameters into a message. Marshaling includes converting the representation of the parameters into a standard format and copying each parameter into the message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a server stub, which de-marshalls(unpack) the parameters and calls the desired server routine using the regular procedure call mechanism.
5. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message. The server stub then hands the message to the transport layer.
6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
7. The client stub de-marshalls the return parameters and execution returns to the caller.

(b)

<u>Qus:</u> What is a stub? How are they generated? State their functionality and purpose.

<u>Ans:</u>

Stub: In distributed computing, a stub is a piece of code that translates arguments exchanged between the client and server during a remote method call.

Stubs can be created in two different ways:

1. **Manual Generation of Stub:**In the case of manual stub generation, the RPC implementer provides a set of translation functions from which a user can generate their own stubs.This technique is simple to use and can accept a variety of parameter types.

2. **Automatic Generation of Stub:** Client and server interfaces are defined using the Interface Definition Language during the automated production of stubs (IDL). For example, an interface specification specifies whether each argument is input, output, or both; only input arguments must be sent from client to server, while only output components must be copied from server to client. This is the most common method for producing stubs.

Functionalities: The function of the stub is to provide transparency to the programmer-written application code.

1. **On the client-side,** the stub handles the interface between the client's local procedure call and the run-time system, marshaling and unmarshalling data, invoking the RPC run-time protocol, and if requested, carrying out some of the binding steps.
2. **On the server-side,** the stub provides a similar interface between the run-time system and the local manager procedures that are executed by the server.

Ans.to.the.qus.no: 4

(a)

Qus: What is causal consistency? Give an example of an application for which causal consistency is the most suitable consistency model.

Ans:

Causal Consistency: Causal consistency implies the idea that causally associated activities should appear in the same sequence on all processes, even if the order of causally independent actions differs between processes.

In the following example, we have a distributed system with four processes: P1, P2, P3, and P4.

At each process, a number of operations are taking place. W(Y,A) indicates that the value of object Y is written as A. The expression R(Y) indicates that the value of object Y is being read.
The following sequence of operations may appear causally consistent at first glance, but it is not. After reading C, P3 must continue to read C or a newer value (possibly B), but cannot return to A.This is because W(Y,C) was conditional on W(Y,A) finishing. We

could say that if P3 had read B on its second read, this system ensured causal consistency.

This is due to the fact that W(Y,B) and W(Y,C) are not causally related. They are, however, concurrent.

(b)

Qus: Differentiate between weak consistency and release consistency. Which of the two will you prefer to use in the design of a DSM system? Give reasons for your answer.

Ans:

Weak consistency meets these criteria:

1. Accesses to synchronization variables associated with a data store are sequentially consistent.
2. No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere
3. Read or write operations on data items are not allowed to be performed until all previous operations to synchronization variables have been performed.

Release consistency is like weak consistency, but there are two operations “lock” and “unlock” for synchronization (“acquire/release” are the conventional names).

1. doing a “lock” means that writes on other processors to protected variables will be known.
2. doing an “unlock” means that writes to protected variables are exported.
3. These variables will be seen by other machines when they do a “lock” (lazy release consistency) or immediately (eager release consistency)

I prefer Release consistency over Weak consistency for two reasons.

1. Advantages of Flexibility and Performance.However, Release consistency's flexibility and efficiency come at the cost of synchronization accesses having to be properly defined and designated as acquisitions and releases.Synchronization accesses, unlike poor consistency, cannot be determined just by instruction opcodes.As a result, programmers are responsible for appropriately identifying, acquiring, and releasing synchronization accesses.

2. For processor consistency, all processes see writes from each processor in the order they were initiated. Release consistency is more relaxed because it does not enforce the ordering between stores that happens in processor consistency. It doesn't follow programmers' intuition as it is relatively less restrictive to compiler optimizations.

(c)

<u>Qus:</u> Why does the simple LRU policy often used for replacing cache lines in a buffer cache not work well as a replacement policy for replacing blocks in a DSM system?

<u>Ans:</u>

LRU is theoretically viable, but it is not inexpensive. To correctly implement LRU, a linked list of all pages in memory must be stored, with the most recently used page at the front and the least recently used page at the back. The problem is that each memory reference necessitates updating the list.

The main disadvantages are :

1. It demands the implementation of extra Data Structure.
2. The level of hardware assistance is very high.

Thus , LRU policy often used for replacing cache lines in a buffer cache not work well as a replacement policy for replacing blocks in a DSM system.

Ans.to.the.qus.no: 5

(a)

Qus: What is the communication protocol used in RPC? Illustrate different steps of RPC protocol with an appropriate figure.

Ans:

Communication Protocol: Different systems, developed on the basis of remote procedure calls, have different IPC requirements. Based on the needs of different systems, several communication protocols have been proposed for use in RPCs.

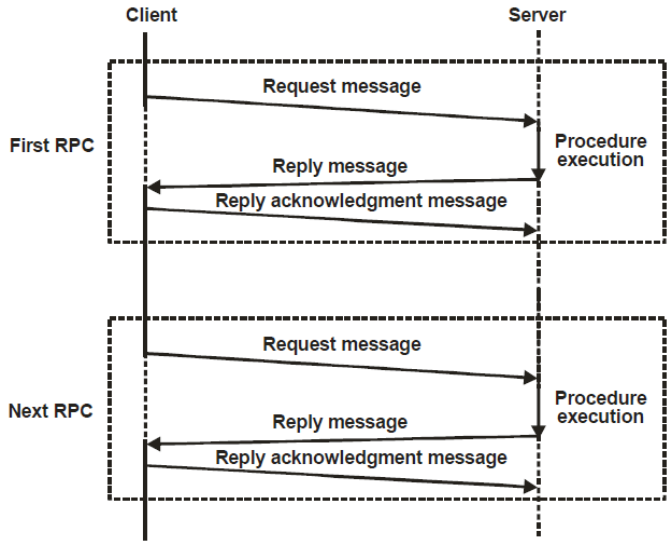


Fig: Request / Reply / Acknowledge reply (RRA) protocol

RRA protocol : The RRA protocol requires that the unique message identifiers associated with request messages be sorted before they can be used. Each reply message includes the message identification from the associated request message, and each acknowledgement message includes the same message identifier. This aids in the matching of a response to its associated request and an acknowledgement to its related response. A client acknowledges a reply message only if it has received responses to all prior requests that relate to this reply. As a result, an acknowledgement message is understood as acknowledging receipt of all reply messages with lower message identifiers that match to the request messages.

(b)

Qus: How can the RRA protocol be redefined for concurrent access to multiple servers? Illustrate with appropriate figures.

Ans:

One of the following three approaches may be used for providing Concurrent Access to Multiple Servers:

1. The use of threads in the implementation of a client process where each thread can independently make remote procedure calls to different servers. This method requires that the addressing in the underlying protocol is rich enough to provide correct routing of responses
2. Another method is the use of the early reply. In this method a call is split into two separate RPC calls, one passing the parameters to the server and the other requesting the result. In reply to the first call, the server returns a tag that is sent back with the second call to match the call with the correct result. The client decides the time delay between the two calls and carries out other activities during this period, possibly making several other RPC calls. A drawback of this method is that the server must hold the result of a call until the client makes a request for it. Therefore, if the request for results is delayed, it may cause congestion or unnecessary overhead at the server.
3. The third approach, known as the call buffering approach, was proposed by Gimson [1985]. In this method, clients and servers do not interact directly with each other. They interact indirectly via a call buffer server. To make an RPC call, a client sends its call request to the call buffer server, where the request parameters together with the name of the server and the client are buffered. The client can then perform other activities until it needs the result of the RPC call. When the client reaches a state in which it needs the result, it periodically polls the call buffer server to see if the result of the call is available, and if so, it recovers the result. On the server side, when a server is free, it periodically polls the call buffer server to see if there is any call for it. If so, it recovers the call request, executes it, and makes a call back to the call buffer server to return the result of execution to the call buffer server.

(c)

Define callback RPC and lightweight RPC.

Callback RPC: In the usual RPC protocol, the caller and callee processes have a client – server relationship. Unlike this, the callback RPC facilitates a peer-to-peer paradigm among the participating processes. It allows a process to be both a client and a server. Callback RPC facility is very useful in certain distributed applications.

Lightweight RPC: Lightweight Remote Procedure Call is a communication facility designed and optimized for cross-domain communications in microkernel operating systems. For achieving better performance than conventional RPC systems, LRPC uses the following four techniques: simple control transfer, simple data transfer, simple stubs, and design for concurrency.

Ans.to.the.qus.no: 6

(a)

Qus: Draw a schematic diagram to represent a distributed shared memory (DSM).

Ans:

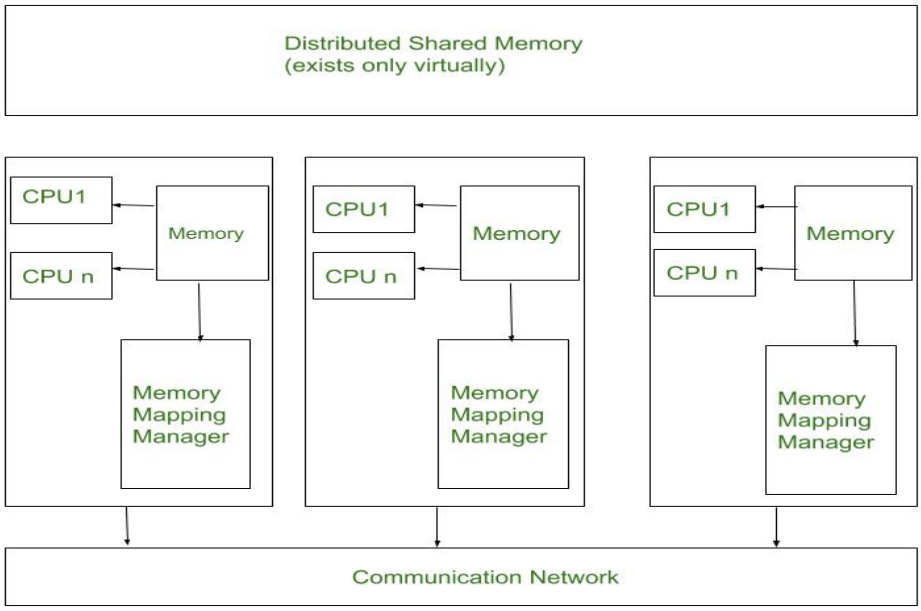


Fig : Distributed shared memory(DSM)

(b)

Qus: What is false sharing ? What should be done to minimise the false sharing problem?

Ans:

False sharing is a data sharing problem,occurring when two different processes access two unrelated variables that reside in the same data block. It may lead to a thrashing problem.

The following technique can be used to minimise the false sharing problem:

- Avoid writing to global data that is accessed from multiple threads.
- Align shared global data to cache line boundaries.
- Don't store temporary, thread specific data in an array indexed by the thread id or rank.
- When parallelizing an algorithm, partition data sets *along* cache lines, not across cache lines.

(c)

Qus: Discuss the relative advantages and disadvantages of using the NRNMB, NRMB, RMB and RNMB strategies in the design of a DSM system.

Ans:

Advantages of NRNMB :

1. Simplest strategy for implementing a sequentially consistency DSM system.
2. Method is simple and easy to implement

Disadvantages of NRNMB :

1. Serializing data access creates a bottleneck.
2. Parallelism,which is a major advantage of DSM is not possible with this method.

Advantages of NRMB :

1. No communications costs are incurred when a process accesses data currently held locally.
2. It allows the applications to take advantage of data access locality.

Disadvantages of NRMB :

1. It is prone to thrashing problem
2. The advantages of parallelism can be availed in this method also.
3. It does not scale well

Advantages of RMB :

1. Simpler abstraction
2. Better portability of Distributed application programs
3. Better performance of some application
4. On demand data moment
5. Large memory space

Disadvantages of RMB :

1. Block size selection becomes a complicated task
2. The mechanism is not fully transparent
3. May lead to wastage of memory due to fragmentation.

Ans.to.qus.no: 7

(a)

UNIX semantics enforces an absolute time ordering on all operations and ensures that every read operation on a file sees the effects of all previous write operations performed on that file [Fig.(a)]. In particular, writes to an open file by a user immediately become visible to other users who have this file open at the same time.

The UNIX semantics is commonly implemented in file systems for single processor systems because it is the most desirable semantics and also because it is easy to serialize all read/write requests. However, implementing UNIX semantics in a distributed file system is not an easy task. One may think that this semantics can be achieved in a distributed system by disallowing files to be cached at client nodes and allowing a shared file to be manage by only one file server that processes all read and write requests for the file strictly in the order in which it receives them. However, even with this approach, there is a possibility that, due to network delays, client requests from different nodes may arrive and get processed at the server node in an order different from the actual order in which the requests were made. Furthermore, having all fie access requests processed by a single server and disallowing caching on a client nodes is not desirable in practice due to poor performance, poor scalability, and poor reliability of the distributed file system.

(b)

General design principle of distributed file system:

· **Clients have cycles to burn:** This principle says that, if possible, it is always preferable to perform an operation on a client's own machine rather than performing it on a server machine. This is because a server is a common resource for all clients, and hence cycles of a server machine are more precious than the cycles of client machines.

· **Cache whenever possible:** Better performance, scalability, user mobility, etc autonomy motivate this principle. Caching of data at clients' sites frequently to improve overall system performance because it makes data available wherever it is being currently used, thus saving a large amount of computing time and network bandwidth. Caching also enhances scalability because it reduces contention on centralized resources.

· **Exploit usage properties :** This principle says that, depending on usage properties (access and modification patterns), files should be grouped into a small number of easily identifiable classes, and then class specific properties should be exploited for independent optimization for improved performance.

· **Minimize system-wide knowledge and change:** This principle is aimed at enhancing the scalability of design. The larger is a distributed system, the more difficult it is to be aware at all times of the entire state of the system and to update distributed or replicated data structures in consistent manner. Therefore monitoring or automatically updating of global information should be avoided as far as practicable.

· **Trust the fewest possible entities:** This principle is aimed at changing the security of the system. For example, it is much simpler to ensure security based on the integrity of the much smaller number of servers rather than trusting thousands of clients. In this case, it is sufficient to only ensure the physical security of these servers and the software they run.

· **Batch if possible:** Parching often helps in improving performance greatly. For example, grouping operation together can improve throughput, although it is often at the cost of latency. Similarly transfer of data across the network in large chunks rather than as individual pages in much more efficient. The full file transfer protocol is an instance of the application of this principle.

(c)

Cache validation is done in two ways:

1) Client initiated approach:

Here client checks for new updates before it accesses its data or it goes with the periodic checking mechanism i.e. client checks for updates after regular intervals of time. Here the pull mechanism is implemented where the client pulls for updates.

2) Server initiated approach:

Here the server is responsible for sending periodic updates to all its clients. The Push protocol is user where she server pushes the new updates to all its clients.

Ans.to.the.qus.no: 8

(a)

Cloud computing technology gives users access to storage, files, software, and servers through their internet-connected devices: computers, smartphones, tablets, and wearables. Cloud computing providers store and process data in a location that's separate from end users.

Essentially, cloud computing means having the ability to store and access data and programs over the internet instead of on a hard drive. This means businesses of any size can harness powerful software and IT infrastructure to become bigger, leaner, and more agile, as well as compete with much larger companies.

Pros and cons of cloud computing is shown below.

Pros:

- **Lower operational costs.** The cloud vendor assumes many equipment and software management tasks, from servers and networking gear to cloud storage. That also includes applying software updates and security patches.
- **Increased IT resources.** Enterprises can access more resources for internal service development and digital transformation projects that directly support business units.

- **Convenient, rapid access to technology.** Enterprises can work with the latest hardware and software, such as new CPUs and GPUs, machine learning and AI applications and network interfaces -- often before they're made available to enterprise buyers.
- **Faster connectivity.** Cloud providers invest in the latest network interface cards and switches, along with multi-Gbps circuits to internet exchange points. This provides the fastest access to data and applications both within the data center and to customers.
- **Greater scale.** The public cloud is engineered for massive scale. Providers can easily expand resource capacity for individual services to meet customers' workload demands.
- **Greater expertise.** Few organizations possess internal expertise in secure infrastructure and security engineering to match what cloud providers offer.
- **More reliable infrastructure.** Cloud providers' physical infrastructure far outstrips what most companies can afford to build or operate. Cloud customers also can access multiple cloud locations, which simplifies redundant deployments. Some cloud services offer built-in multisite redundancy.

Cons:

- **A complicated shared security model.** Security policies and management are split between the provider and user. Understanding the division in this shared responsibility is crucial, because mistakes can expose vast amounts of sensitive data.
- **Complex pricing structures.** Some services, such as compute instances, have multiple subscription tiers and pricing schemes. These variables make pricing and total cost of ownership analysis tedious and time-consuming; doing so typically requires software assistance from built-in or third-party tools.
- **Outbound data transfer costs**. It is expensive to access large data sets, and this also creates a disincentive for an organization to move from one cloud provider to another.
- **Less flexibility than DIY environments.** Many configuration choices are made by the provider, so customers have limited control.
- **Sketchy, inconsistent customer support.** Cloud services providers can be difficult to reach or slow to respond to technical issues or cost concerns. As a result, many organizations contract with a third-party cloud management and support partner.

- **Fast, redundant connectivity**. Cloud computing requires either reliable connections to networks and the internet or a direct private link to the provider. This is especially important for remote locations.
- **Cloud-specific skills**. Most internal IT organizations do not possess the cloud design and operations expertise found on a cloud provider's payroll. Such cloud-skilled staff can be hard to recruit and retain, as workers with those advanced skills are attractive to other organizations as well as to the cloud providers themselves.
- **Country- or industry-specific regulatory requirements.** Organizations must plan carefully, especially when data and workloads are hosted outside one's residence or country with strict privacy laws. Note that a cloud provider's presence in a particular location may imply jurisdiction, and a need to comply with local regulations.

(b)

As for the cloud service, I would use PaaS for developing and launching my e-commerce website. Because Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a cloud service provider on a pay-as-you-go basis and access them over a secure Internet connection. Like IaaS, PaaS includes infrastructure—servers, storage, and networking—but also middleware, development tools, business intelligence (BI) services, database management systems, and more. PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing, and updating.

And for deployment model, I'd use public cloud. Because the public cloud makes it possible for anybody to access systems and services. The public cloud may be less secure as it is open for everyone. The public cloud is one in which cloud infrastructure services are provided over the internet to the general people or major industry groups. The infrastructure in this cloud model is owned by the entity that delivers the cloud services, not by the consumer. It is a type of cloud hosting that allows customers and users to easily access systems and services. This form of cloud computing is an excellent example of cloud hosting, in which service providers supply services to a variety of customers. In this arrangement, storage backup and retrieval services are given for free, as a subscription, or on a per-use basis.

(c)

Scalability and multitenancy are the aspects of cloud computing that makes it useful for Prognostics and Health Management (PHM) applications. A prognosis approach based on the Cloud Computing model and the principle of multitenancy in order to present the Prognosis as a Service can be developed using these aspects of cloud. This approach provides an effective prognosis solution at the request of a client while ensuring a better quality of service. The effectiveness of our solution depends on the criteria for the performance of the prognosis system based on accuracy, accuracy, mean squared error and a Quality of Service (QoS).