# Parallel and Distributed Computing CS3006 (BDS-6A) Lecture 14

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

16 March, 2023

## Previous Lecture

- Distributed Systems – issues
- Operating Systems:
  - Distributed, Network, Middleware
- Distributed Systems – challenges
- Need for High Performance Scalability
- Clusters
  - High Availability/failover clusters
  - Load Balancing
  - High Performance

## High Performance Cluster

- Donald Becker of NASA assembled such a cluster in 1994
- It is also known as the *Beowulf cluster*.
- Essentially, any *group of Linux machines* dedicated to a *single purpose* with a *centralized node* (for coordination) can be called a *Beowulf cluster*.
- These types of clusters *increase performance* and *scalability* for applications, particularly *computationally or data intensive tasks*.
- Applications of such systems include *data mining*, *simulations*, *parallel processing*, *weather modeling*, etc.

## Beowulf Cluster

- In most cases *client nodes do not have keyboards or monitors*, and are *accessed* only *via remote login* or possibly *serial terminal*.
- Beowulf nodes can be thought of as a *CPU + memory package* which can be *plugged* into the *cluster*, just like a CPU or memory module can be plugged into a motherboard.
- Commonly used parallel processing libraries include *Message Passing Interface (MPI)* and *Parallel Virtual Machine (PVM).*
- These (MPI, PVM) permit the programmer to *divide a task* among a group of *networked computers*, and *collect the results* of processing.

# Building your own Beowulf Cluster

- Hundreds of examples of *Beowulf clusters* are already linked from the Beowulf Project site (www.beowulf.org).
- THE SPACE A small cluster of two to four nodes can fit on a small table, so not much space is required.
- HARDWARE For small clusters, desktop tower systems, even used PCs that are too slow for recent Windows programs are fine. You will also need a hub or switch to tie the machines together.
- SOFTWARE Linux on each node + message-passing software (PVM or MPI) Beowulf.org links to everything you need, and most of it's free.

Some History: http://yclept.ucdavis.edu/Beowulf/aboutbeowulf.html

## Benefits of Cluster Computing

- Processing Power: The parallel processing power of a high-performance cluster can, in many cases, prove *more cost effective* than a mainframe with similar power.

- Scalability: Perhaps the greatest advantage of computer clusters is the *scalability* they offer. While mainframe computers have a fixed processing capacity, computer *clusters can be easily expanded* as requirements change by adding additional nodes to the network.

- Availability: When a mainframe computer fails, the *entire system fails*. However, if a node in a computer cluster fails, its operations can be *continued by other nodes*.

## Benefits of Cluster Computing

- Reduced Cost: The price of *off-the-shelf consumer desktops* has *plummeted* in recent years, and this drop in price has corresponded with a vast increase in their processing power and performance. The average desktop PC today is many times more powerful than the first mainframe computers.

- Improved Network Technology: Driving the development of computer clusters has been a *vast improvement in the technology related to networking*, along with a reduction in the price of such technology. Computer clusters are typically connected via a single virtual local area network (VLAN), and the network treats each computer as a separate node.
  - Information can be passed throughout these networks with very little lag, ensuring that data doesn't bottleneck between nodes.

## Challenges in Cluster Computing

- A cluster should be a *single computing resource* and provide a single *system image*.

- The supporting *operating system* and *communication mechanism* must be efficient enough to remove any *performance bottlenecks*.

- It *must provide scalability* by letting the system scale up or down. The scaled-up system should provide more functionality or better performance. The system's total computing power *should increase proportionally* to the increase in resources. The main motivation for a scalable system is to provide a flexible, cost effective information-processing tool.

## Issues to be considered

- Cluster networking: If you are mixing hardware that has *different networking technologies*, there will be large differences in the *speed* with which data will be accessed and how individual nodes can communicate.
  - If the budget allows, it is better if all of the machines to be included in the cluster have *similar networking capabilities*, and if at all possible, have *network adapters* from the same *manufacturer*.

- Cluster Software: One has to build versions of clustering software for each kind of system one includes in a cluster.

## Issues to be considered

- Timing: This is the most problematic aspect of cluster. Since these machines have *different performance profile* our code will execute at *different rates* on the *different kinds of nodes*. This can *cause serious bottlenecks* if a process on one node is waiting for results of a calculation on a slower node.

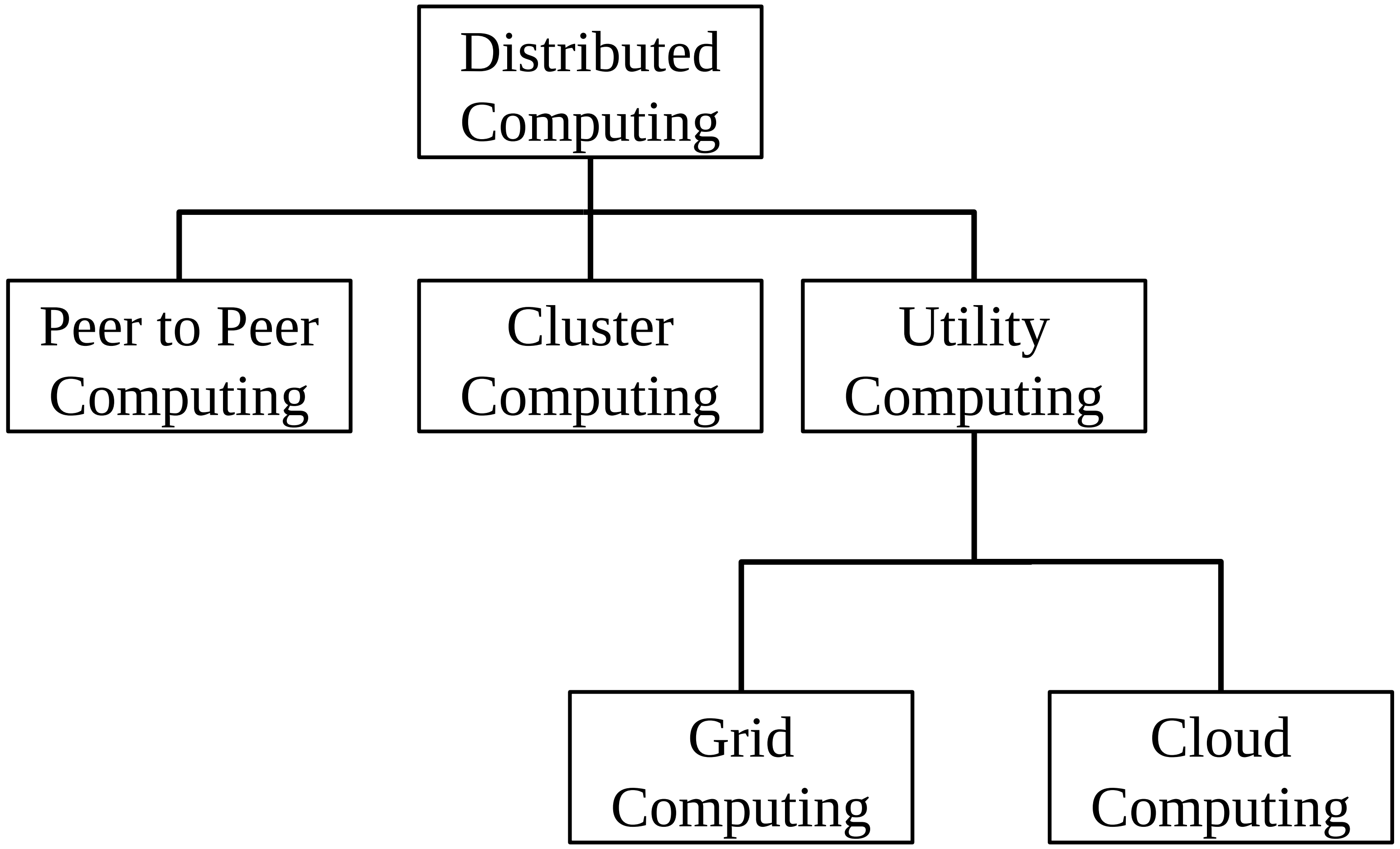
- Programming: Our code will have to be written to support the *lowest common denominator* for data types supported by the *least powerful node* in our cluster. With mixed machines, the more powerful machines will have attributes that cannot be attained in the least powerful machine.

## Issues to be considered

- Network Selection: There are a number of different kinds of network topologies, including *buses*, *cubes* of various degrees, and *grids/meshes*. These network topologies will be implemented by use of one or more network interface cards, or *NICs*, installed into the head-node and compute nodes of our cluster.

- Speed Selection: No matter what topology you choose for your cluster, you will want to get the *fastest network that your budget allows*. Fortunately, the availability of high speed computers has also forced the development of *high speed networking systems*. Examples are: *10Mbit Ethernet*, *100Mbit Ethernet*, *Gigabit networking*, *channel bonding,* etc.

# Distributed Computing



## Clusters

- Cluster is tightly coupled, whereas a Grid or a cloud is loosely coupled
- All nodes work cooperatively together as a single integrated computing resource so conceptually it is smashing up many machines to make a powerful machine

- Types of clusters
  - High Availability clusters
  - Load balancing clusters
  - High performance Computing clusters

## Utility Computing

- Analogy is derived from the real world
  - service providers supply utility services, electrical power, gas, and water to consumers
  - Consumers in turn pay service providers based on their usage

- Users (consumers) pay providers for using computing power only when they need to use it

## Grid Computing

- What is a grid?
- Collection of multiple resources to reach a common goal

- Electricity grid
    - A network of synchronized power providers and consumers that are connected by transmission and distributed lines.
    - “The power grid” links together power plants of many different kinds
    - Users get access to electricity without any care from where and how the electricity is actually generated

## Computer Grid

“An infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations.”

“Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed “autonomous” resources dynamically at runtime depending on their availability, capability, performance, cost, and users’ quality -of-service requirements.”

## Grid Computing

- Provides…
  - resource sharing (processing, data storage, applications access, etc.)
  - Location transparency

- In other words, “Grid" links together computing resources (PCs, workstations, servers, storage) and provides the mechanism needed to access them

- Allows users to use more resources than they independently own

## Practical application

- Practicality depends on the type of applications which includes

- High throughput problems which involves computing grids to schedule tasks across resources

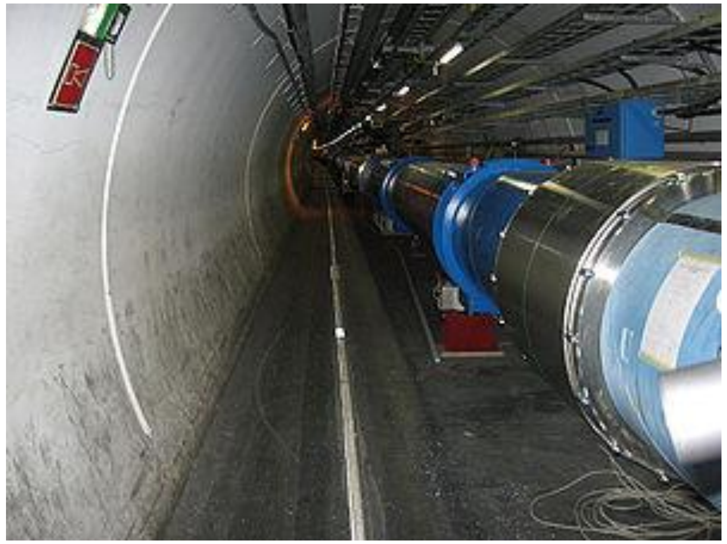
- Embarrassingly parallel problems where the task can be broken down into parts which are independent of each other

## Need for Grid Computing

- A systems' CPU cycles are wasted when not in use which can be used efficiently by combining servers, storage devices and other networks into a single virtual system to share resources dynamically
- Industries like bio-medical field, finance, oil exploration and motion picture require massive CPU capacity
- Grid computing provide better utilization of resources with parallel CPU capacity

## Examples

- The Large Hadron Collider (LHC) is the world's largest and most powerful particle collider, and the largest single machine in the world,[1] built by the European Organization for Nuclear Research (CERN) from 1998 to 2008
- A particle accelerator is a device that uses electromagnetic fields to propel charged particles to high speeds and to contain them in well-defined beams



## Large Hadron Collider

- As per Wikipedia,

  “The LHC was built in collaboration with over 10,000 scientists and engineers from over 100 countries, as well as hundreds of universities and laboratories. It lies in a tunnel 27 kilometres (17 mi) in circumference, as deep as 175 metres (574 ft) beneath the Franco-Swiss border near Geneva, Switzerland”

## What is eSceince

“It is computationally intensive science that is carried out in highly distributed network environments, or science that uses immense data sets (that require grid computing)”

- Collaborations
- High Cost Infrastructure – particle accelerator
- Increased Research Parameters
- Data Generated is massive and distributed
- Increased Computational power at the individual and collective level
- Creation of a computational infrastructure by coupling wide-area distributed resources such as databases, storage servers, high-speed networks, supercomputers and clusters for solving large-scale problems, leading to what is popularly known as Grid computing

## Interplanetary Grid

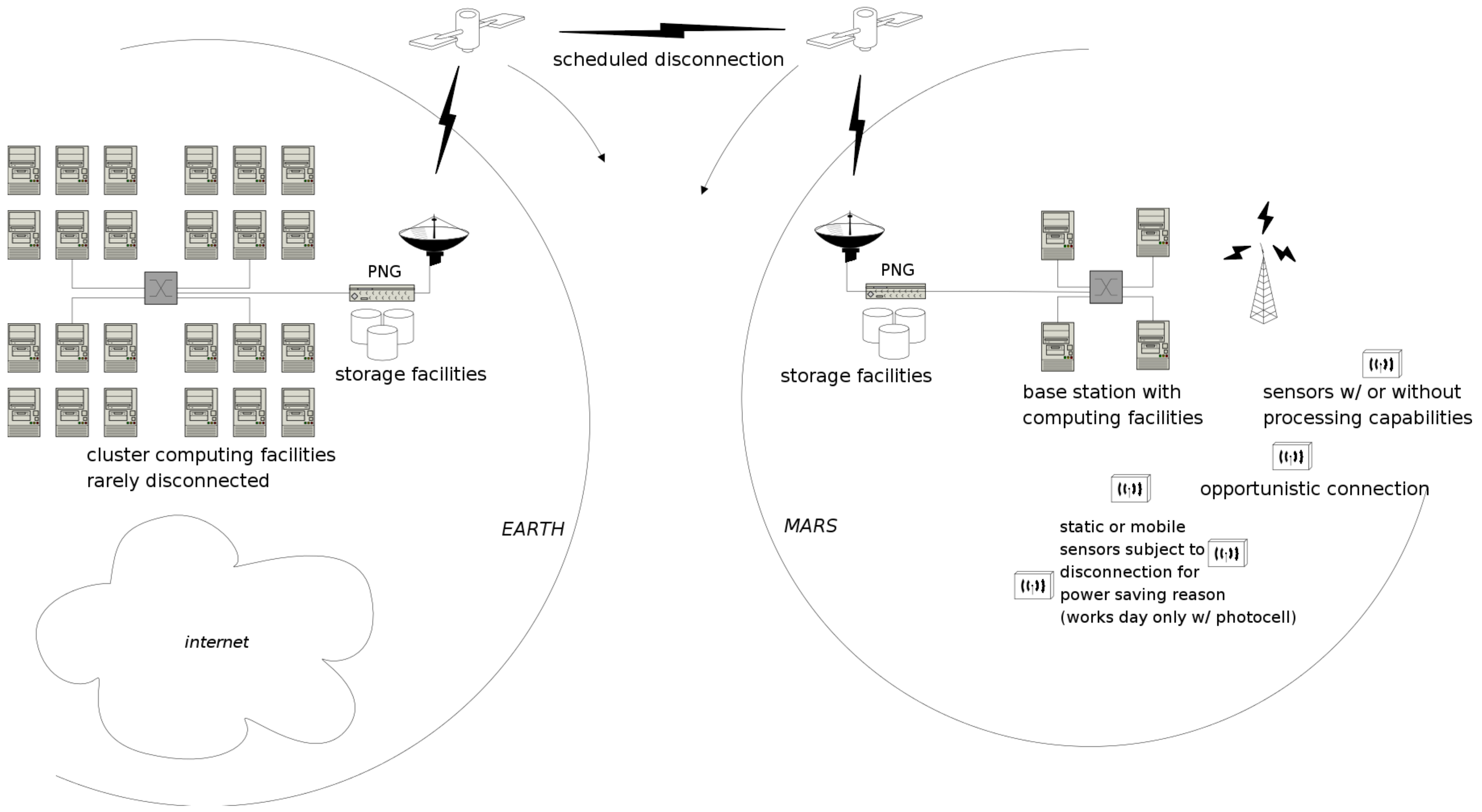
- Space missions require computing/storage resources to process collected data (from robots, cameras, sensors...)

- Sending large computing equipment on remote planets :

too expensive!

- Need for a computing Interplanetary Grid which can support space challenges and provide an unified framework for computing collected data.

# Interplanetary Grid



## Interplanetary Grid

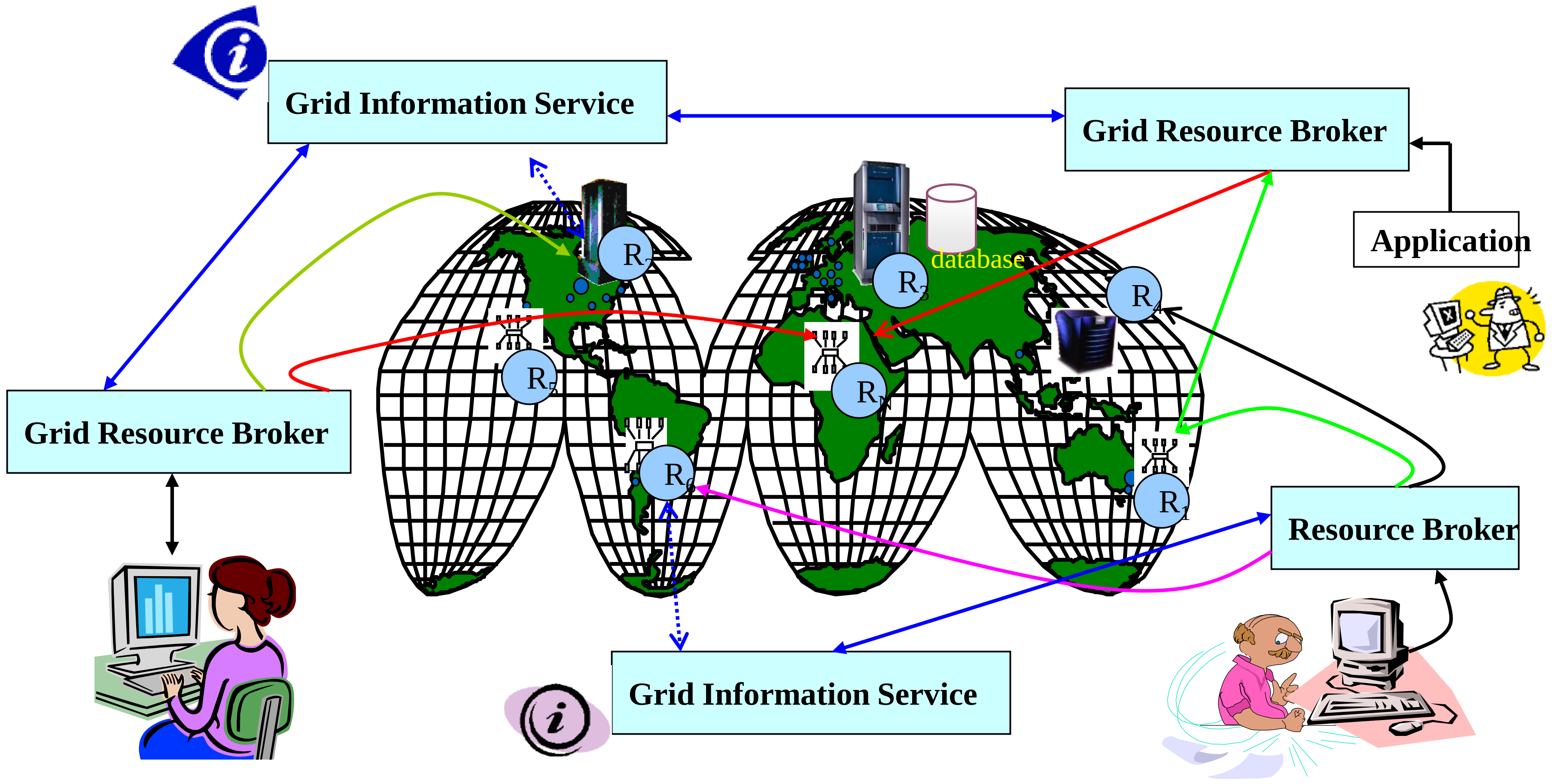
- Infrastructure
  - Derived from Interplanetary networks
  - Heavy computing resources on Earth
  - Lightweight computing remote resources
- Services
  - Remote intervention without human
  - Ultra long latencies networks
  - Disruptive connections
- Applications
  - Supporting space missions applications with local and remote resources

IPG = Grid + Autonomic Gateways + DTN

## Major Concerns

- Major concerns of researchers and developers while setting up the Grid are
  - Improving distributed management whilst retaining full control over locally managed resources
  - Improving the availability of data and identifying problems and solutions to data access patterns
  - Providing researchers with a uniform user-friendly environment that enables access to a wider range of physically distributed facilities improving productivity

# A Typical Grid Computing Environment



## Grid Middleware

- Grid resources are registered within one or more Grid information services.
- The end users submit their application requirements to the Grid resource broker which then discovers suitable resources by querying the Information services schedules the application jobs for execution on these resources
- Monitors their processing until they are completed
- Other services
  - **Security**
  - **Information, directory**
  - **Resource allocation**
  - **Application development, execution management and scheduling**
  - **Resource aggregation**
- Software tools and services providing these capabilities to link computing capability and data sources in order to support distributed analysis and collaboration are collectively known as Grid middleware

## Challenges

- Heterogeneity
- Dispersed resource
    - Handling of Grid resources that are spread across political and geographical boundaries and are under the administrative control of different organizations
    - Unpredictable Availability

## Middleware Development Architectures

- Architecture based on Virtual Organizations

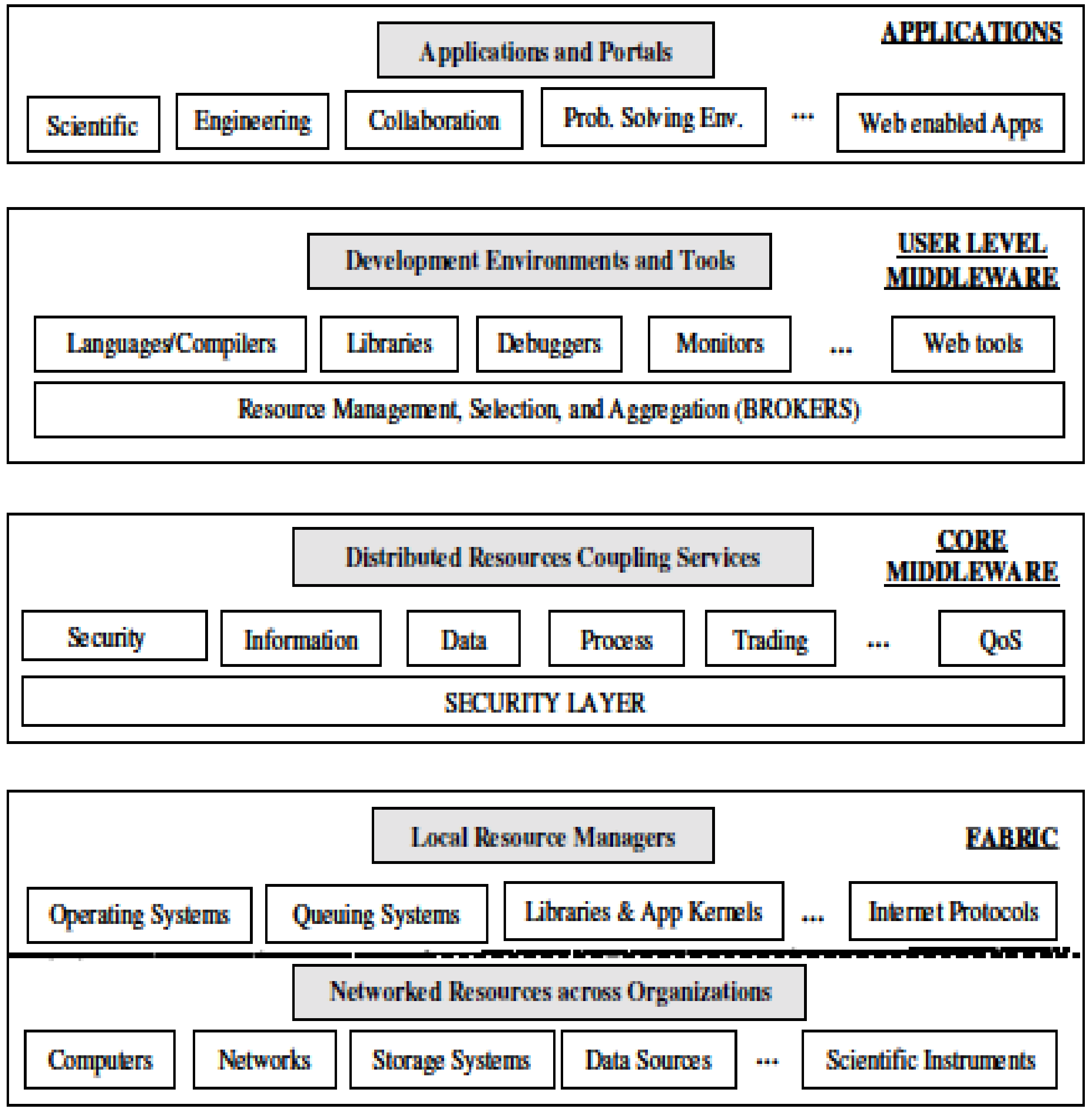
  "A VO defines the resources available for the participants and the rules for accessing and using the resources. Within a VO, participants belonging to member organizations are allocated resource share based on urgency and priority of a request as determined by the objectives of the VO"

- Another complimentary Grid architecture is based on economic principles in which resource providers compete to provide the best service to resource consumers who select appropriate resources based on their specific requirements, the price of the resources and their expectations of Quality-of-Service (QoS) from the providers.
  - For example, QoS terms can be the deadline by which the resource needs to be available and the maximum price (budget) that can be paid by the user for the service.

## Grid Components

- Remote storage and/or replication of data sets
- Publication of datasets using global logical name and attributes in the catalogue
- Security –access authorization and uniform authentication
- Uniform access to remote resources (data and computational resources)
- Publication of services and access cost
- Composition of distributed applications using diverse software components including legacy programs
- Discovery of suitable datasets by their global logical names or attributes
- Discovery of suitable computational resources
- Mapping and Scheduling of jobs (Aggregation of distributed services)
- Submission, monitoring, steering of  jobs execution
- Movement of code/data between the user Desktop machines and distributed resources
- Enforcement of quality of service requirements
- Metering and Accounting of resource usage

## Grid Development Architecture



## Grid Development Architecture

- **<u>Grid Fabric Layer</u> consists of distributed resources such as computers, networks, storage devices and scientific instruments**
  - The computational resources represent multiple architectures such as clusters, supercomputers, servers and ordinary PCs which run a variety of operating systems
  - Scientific instruments such as telescope and sensor networks provide real-time data that can be transmitted directly to computational sites or are stored in a database

- **<u>Core Grid Layer</u> offers services**
  - Remote process management, co-allocation of resources, storage access, information registration and discovery, security, and aspects of Quality of Service (QoS) such as resource reservation and trading
  - **Abstracts the complexity and heterogeneity of the fabric level**
    - Provides a consistent method for accessing distributed resources

## Grid Development Architecture

- **User-level Layer** middleware utilizes the interfaces provided by the low-level layer to provide higher level abstractions and services

  - These include application development environments, programming tools and resource brokers for managing resources and scheduling application tasks for execution on global resources

- **Grid applications and portals** are typically developed using Grid-enabled programming environments and interfaces and brokering and scheduling services provided by user-level middleware

  - Grid portals offer Web-enabled application services, where users can submit and collect results for their jobs on remote resources through the Web.

## Operational Flow

- The users compose their application as a distributed application using application development tools
- The users specify their analysis and quality-of-service requirements and submit them to the Grid resource broker
- The Grid resource broker performs resource discovery and their characteristics using the Grid information service
- The broker identifies resource service prices by querying the Grid market directory
- The broker identifies the list of data sources or replicas and selects the optimal ones
- The broker also identifies the list of computational resources that provides the required application services
- The broker ensures that the user has necessary credit or authorized share to utilize resources.

## Operational Flow Cont…

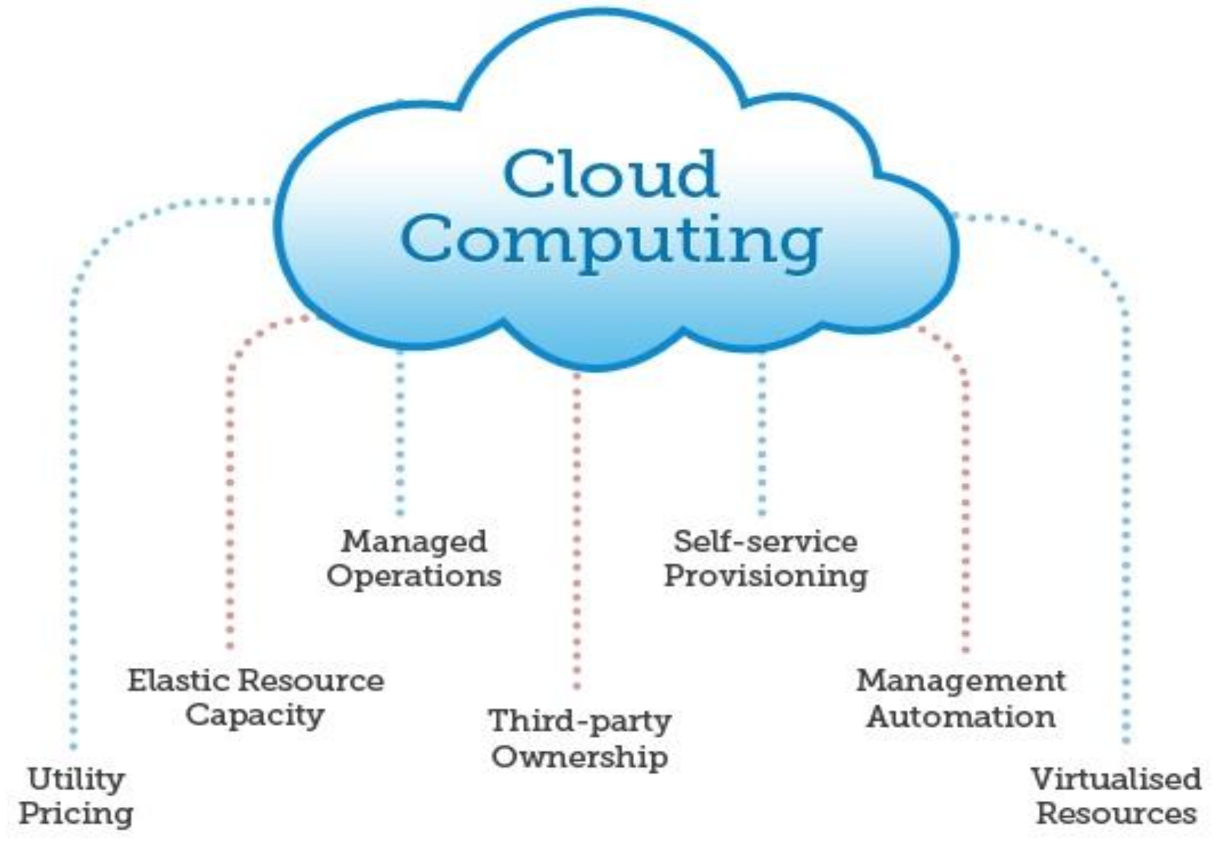
- The broker scheduler maps and deploys data analysis jobs on resources that meet user quality-of-service requirements.
- The broker agent on a resource executes the job and returns results.
- The broker collates the results and passes to the user
- The metering system charges the user by passing the resource usage information to the accounting system
- The accounting system reports resource share allocation or credit utilisation to the user.

## Advantages of Grid Computing

- Exploitation of under utilized resources
- Files and databases can span many systems
- Data can be duplicated for backup
- It is easy to do resource balancing; scheduling can be done on machines with low utilization
- High-end computing system, reliability is increased by the use of expensive hardware
- Redundant communication paths as multiple routes are available

## Cloud Computing

- A pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services that are delivered on demand to external customers over the Internet

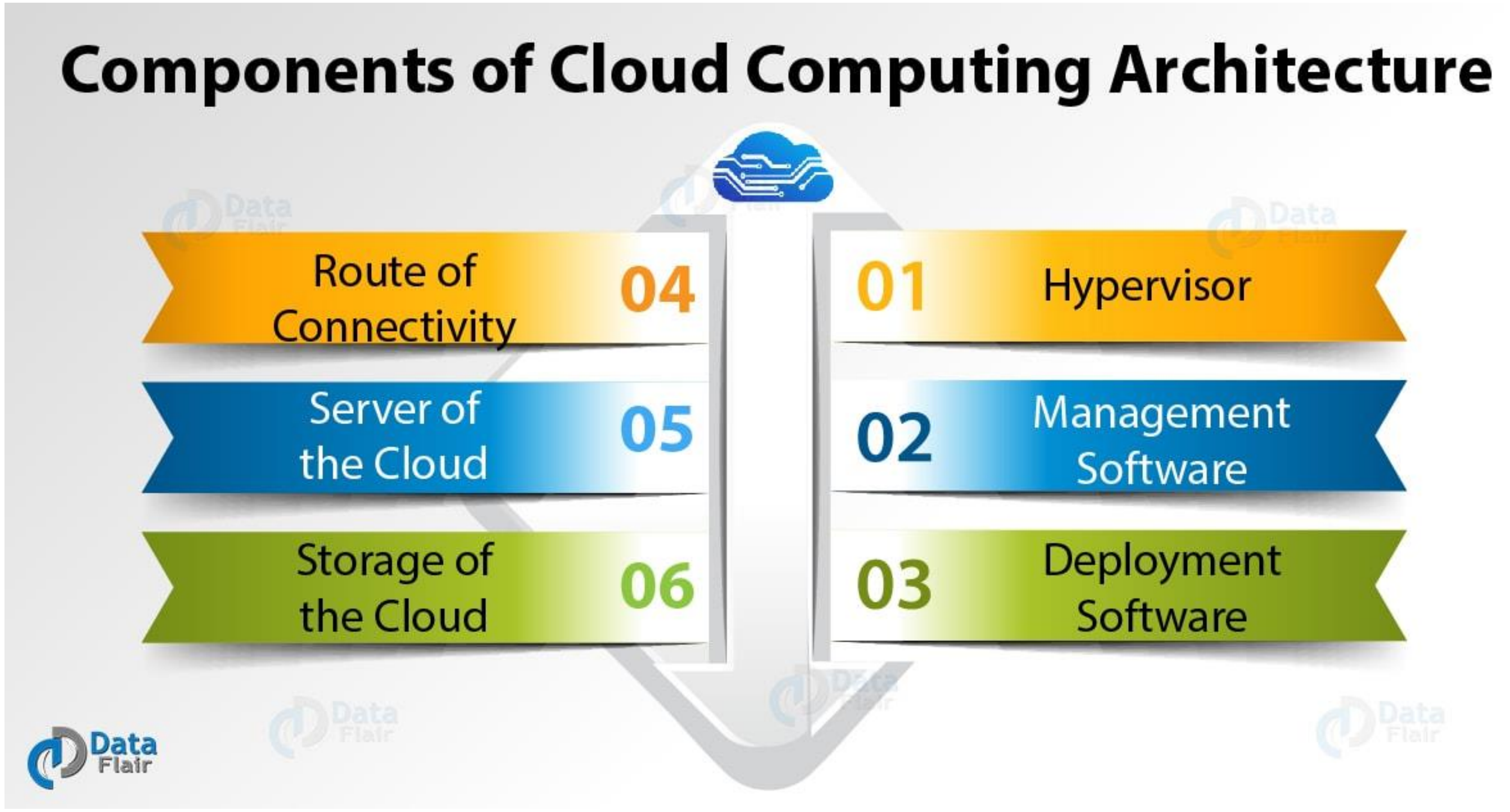


## Cloud infrastructure

- Cloud computing is a computing paradigm that involves
  - outsourcing of computing resources
  - capabilities of expendable resource scalability
  - on-demand provisioning with little or no up-front IT infrastructure investment costs

## Cloud Computing

- Clouds can execute any job that was good for Grids plus
  - More attractive due to platform plus elastic on-demand model
  - Services can be dynamically configured (via virtualization or other approaches)  and delivered on demand
  - Massively  scalable
  - Can be encapsulated as an abstract entity that delivers different levels of service
  - Illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning
  - Ability to pay for use of computing resources on a short-term basis as needed (e.g., processors by the hour and storage by the day) and release them as needed



## Components of Cloud Computing Architecture

## Cloud Components

- Hypervisor (*Virtual Machine Monitor)*

  It is a low-level program that acts as a Virtual Machine Manager. It allows to share the single physical instance of cloud resources between several users. The Hypervisor provides a user with a platform which is known as *Virtual Operating Platform*.

- Management Software

  It consists of various plans and the strategies to manage and improve the performance of the cloud. Features include on-time delivery of storage,  security, all-time access, compliance auditing, disaster management etc.

- Deployment Software

  This deployment consists of all the mandatory installations and configurations of the cloud *to initiate the working of the SaaS, PaaS, and IaaS*.

## Cloud Components

- Cloud Server

  Cloud servers have all the software they need to run and can operate as non-dependent units. It also has the profit because it is incredibly simple and fast to upgrade by adding memory and disk space, further as being more cost-effective.

- Cloud Storage

  It provides access to offsite storage that may be provisioned instantly are versatile and could be scaled automatically at runtime and is globally accessible.

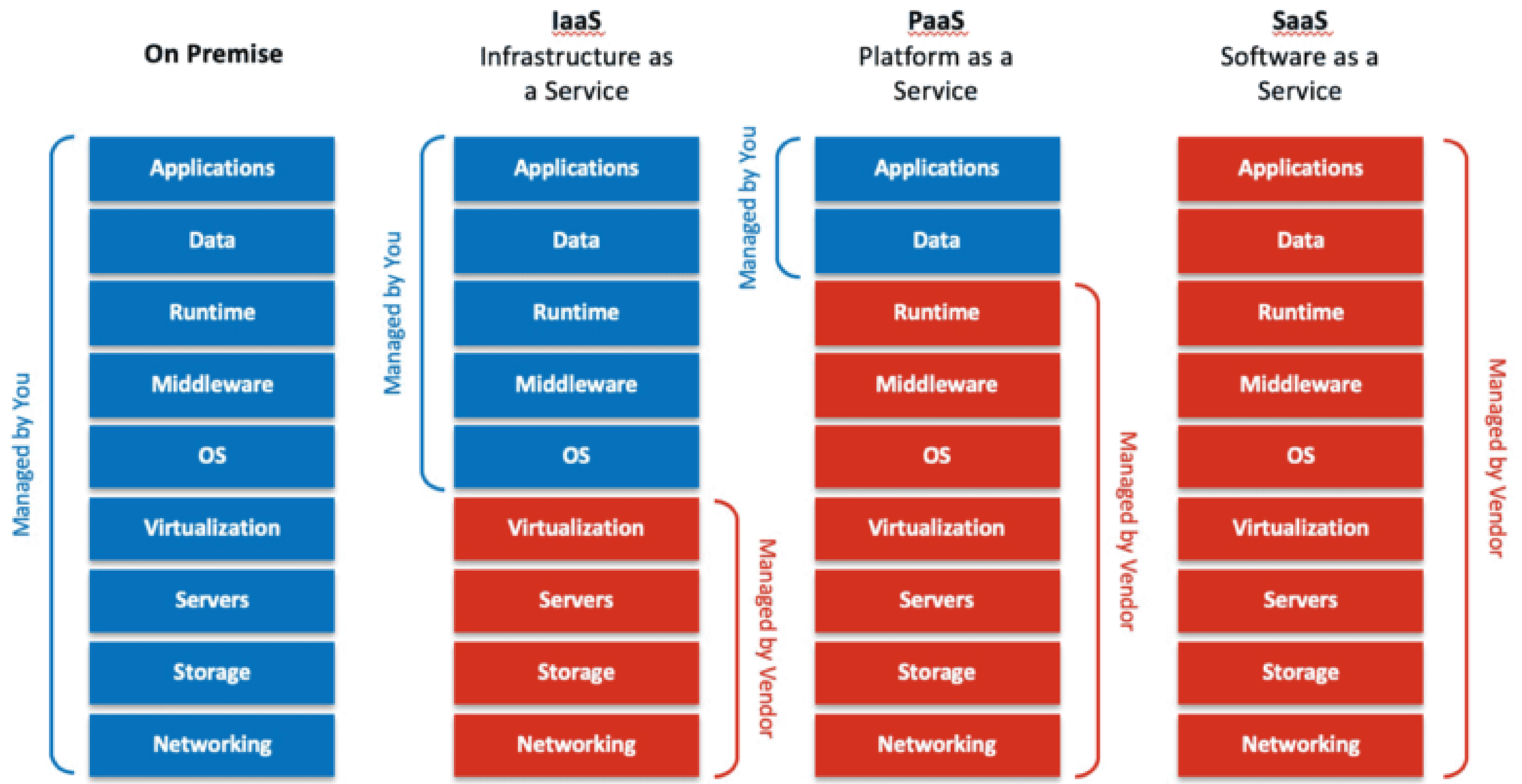
- Route of Connectivity

  The network through which the whole cloud gets connected. There are many cloud servers present which connects with the help of this virtual route. The speed of transfer depends on the network which is the internet connection.

## Cloud Computing Service Model

- Cloud computing offers its benefits through three types of service or delivery models namely
- Infrastructure-as-a-service (IaaS)
  - Hardware software, equipment, can scale up and down
  - Examples: Amazon Elastic Compute Cloud (EC2) and Simple Storage Service (S3)
- Platform-as-a-service (PaaS)
  - High level integrated environment to build, test, and deploy custom apps
  - Example: Google App Engine
- Software-as-a-Service (SaaS)
  - Special purpose software that is remotely accessible
  - Examples: Google Maps, Google Docs, Gmail, Microsoft OneDrive

# Cloud Computing Service Model



## Types of clouds

- Public cloud
  - the cloud infrastructure is accessible to general public
  - cloud resources are accessible via the internet and the provider is responsible for the management of the shared infrastructure
  - Amazon Web Services EC2
- Private cloud
  - the general public does not have access to the private cloud
  - cloud resources in this model may be located within the client organization premises or offsite
- Community cloud
- Hybrid cloud

## Types of clouds

- Community cloud
  - cloud infrastructure is shared by multiple organizations or institutions that have a shared concern or interest
  - both the public and the organizations forming the community cloud have access to the cloud services offered by the community cloud
- Hybrid cloud
  - combines different clouds for example the private and public clouds
  - general public does not have access to the cloud, but the organization uses infrastructure in both the public and private cloud

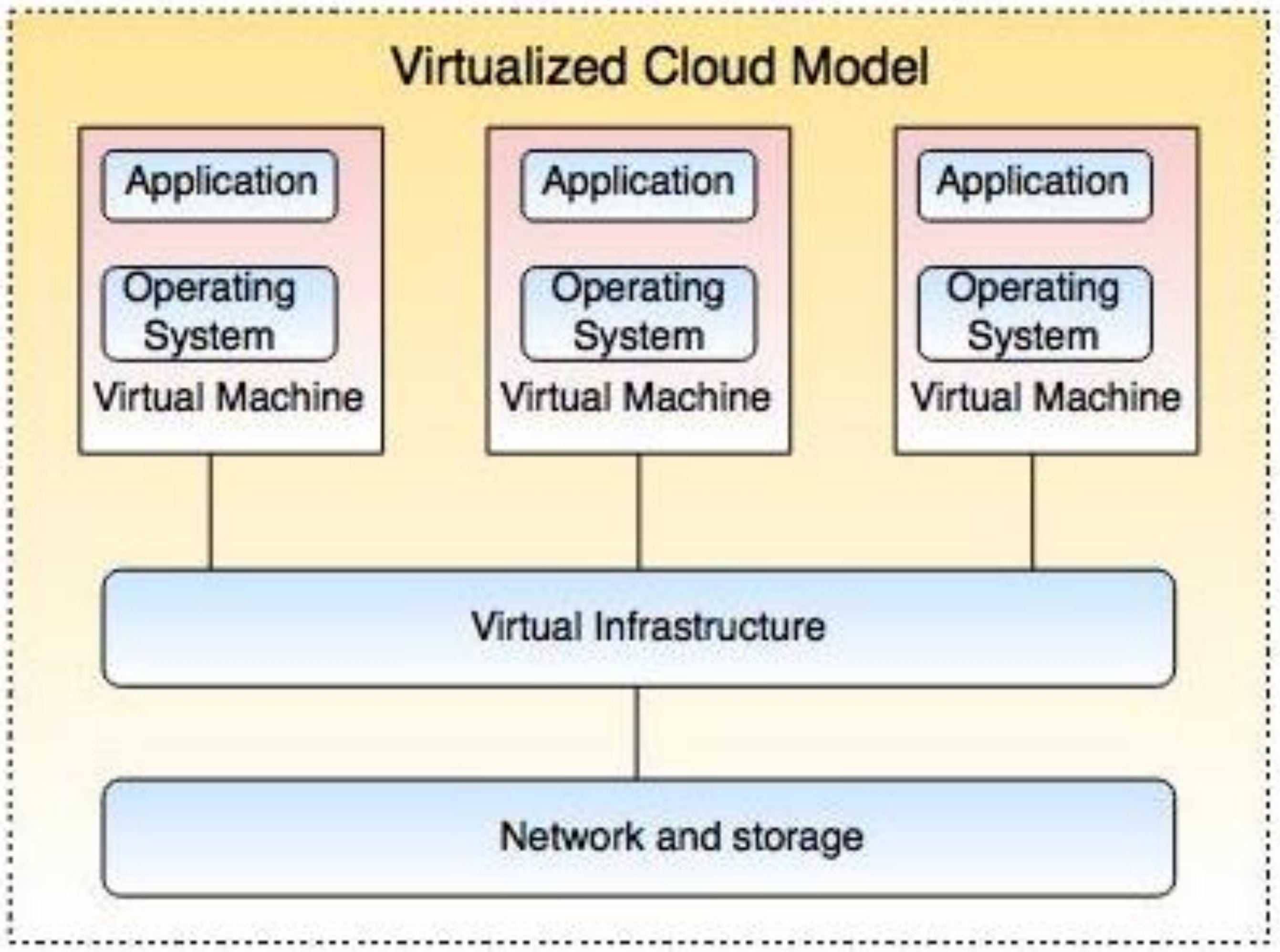
## Virtualization

- Virtualization is the partitioning of a single physical server into multiple logical servers.

- Once the physical server is divided, each logical server behaves like a physical server and can run an OS and applications independently.

- Many popular companies like *VMWare* and *Microsoft* provide virtualization services, where instead of using your personal PC for storage and computation, you use their virtual server. They are fast, cost-effective and less time consuming.

# Virtualization



## Virtualization purposes

- Network Virtualization: It is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others and each channel is independent of others and can be assigned to a specific server or device in real time.

- Storage Virtualization: It is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. Storage virtualization is commonly used in storage area networks (SANs).

- Server Virtualization: Server virtualization is the masking of server resources like processors, RAM, operating system etc, from server users. The intention of server virtualization is to increase the resource sharing and reduce the burden and complexity of computation from users.

## Cloud vs Grid

- Cloud is a collection of computers usually owned by a single party.
- Cloud computing is a centralized model.
- Cloud computing works more as a service provider for utilizing computer resource.

- A grid is a collection of computers which is owned by  multiple parties in multiple locations and connected together so that users can share the combined power of resources.
- Grid computing is a decentralized model, where the computation could occur over many administrative model.
- Grid computing uses the available resource and interconnected computer systems to accomplish a common goal.

# Basic Communication Operations

## Basic Communication Operations

- Preliminaries
  - Exchanging data is a fundamental requirement for most of the parallel algorithms
  - $\boldsymbol{t_s + mt_w}$ - the simplified communication cost model :-
    - Over distributed memory infrastructure
    - Assuming cut-through routing
  - This chapter is about commonly used basic communication patterns over the different interconnections
    - We shall derive communication costs of these operations on different interconnections.

## Basic Communication Operations

- **Assumptions for the Operations**
  - Interconnections support cut-through routing
  - Communication time between any pair of nodes in the network is the same (regardless of the number of intermediate nodes)
  - Links are bi-directional
    - The directly connected nodes can simultaneously send messages of *m words* without any congestion
  - Single-port communication model
    - A node can send on only one of its links at a time
    - A node can receive on only one of its links at a time
  - However, a node can receive a message while sending another message at the same time on the same or a different link.

## Basic Communication Operations

- **One-to-All Broadcast**
- **All-to-One Reduction**

## Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- **One-to-All Broadcast**
  - A single process sends identical data to all other processes.
    - Initially one process has data of size *m*.
    - After the broadcast operation, each of the processes have their own copy of size *m*.
- **All-to-One Reduction**
  - Dual of one-to-all broadcast
  - The *m-sized* data from all processes are combined through an associative operator
  - accumulated at a single destination process into one buffer of size *m*

# Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

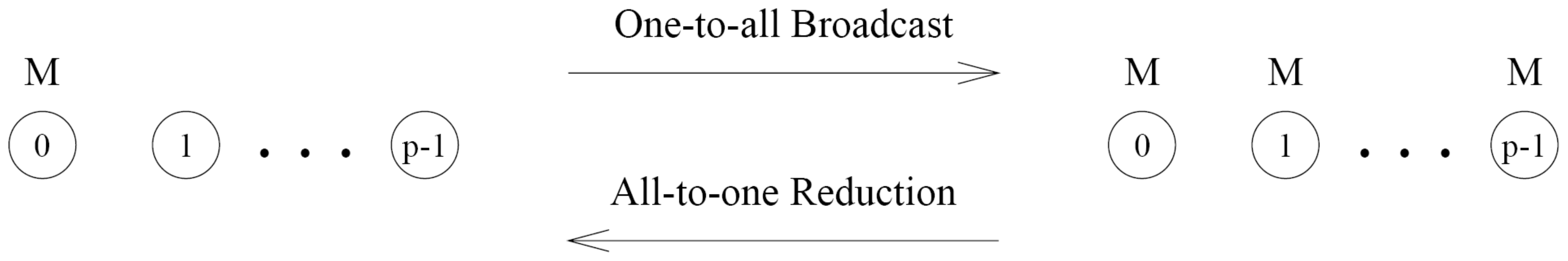


**Figure 4.1** One-to-all broadcast and all-to-one reduction.

# Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- **Linear Array or Ring**
  - Naïve solution
    - Sequentially send *p* - 1 messages from the source to the other *p* - 1 processes
      - Bottlenecks, and underutilization of communication network
    - Solution?
  - Recursive doubling
    - Source process sends the message to another process
    - In the next communication phase both the processes can simultaneously propagate the message

## Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- **Linear Array or Ring**
  - Recursive Doubling Broadcast

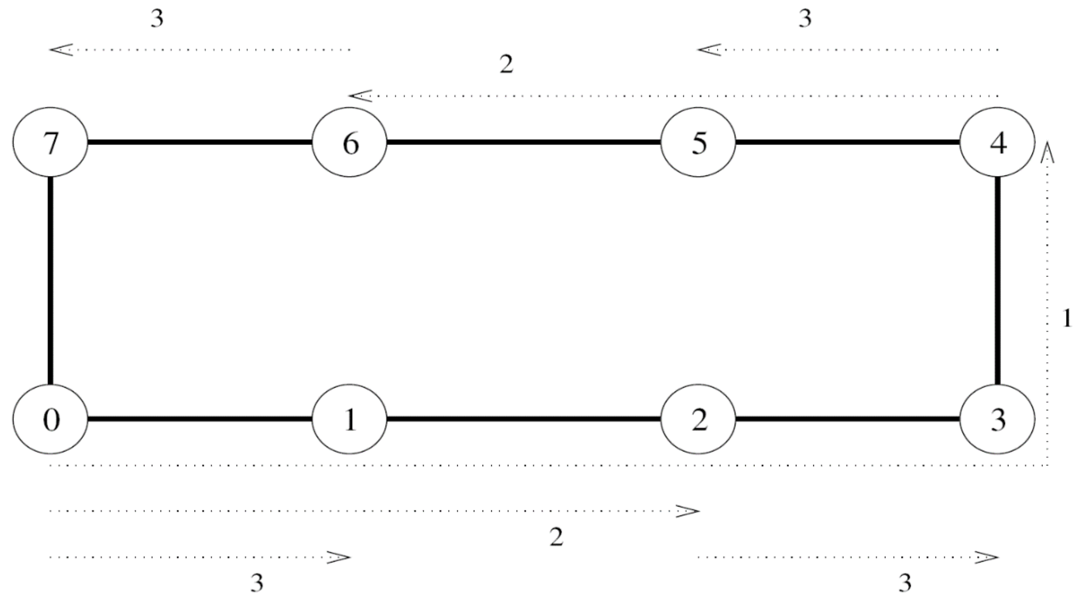


**Figure 4.2** One-to-all broadcast on an eight-node ring. Node 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

## Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- **Linear Array or Ring**
  - Recursive Doubling Reduction

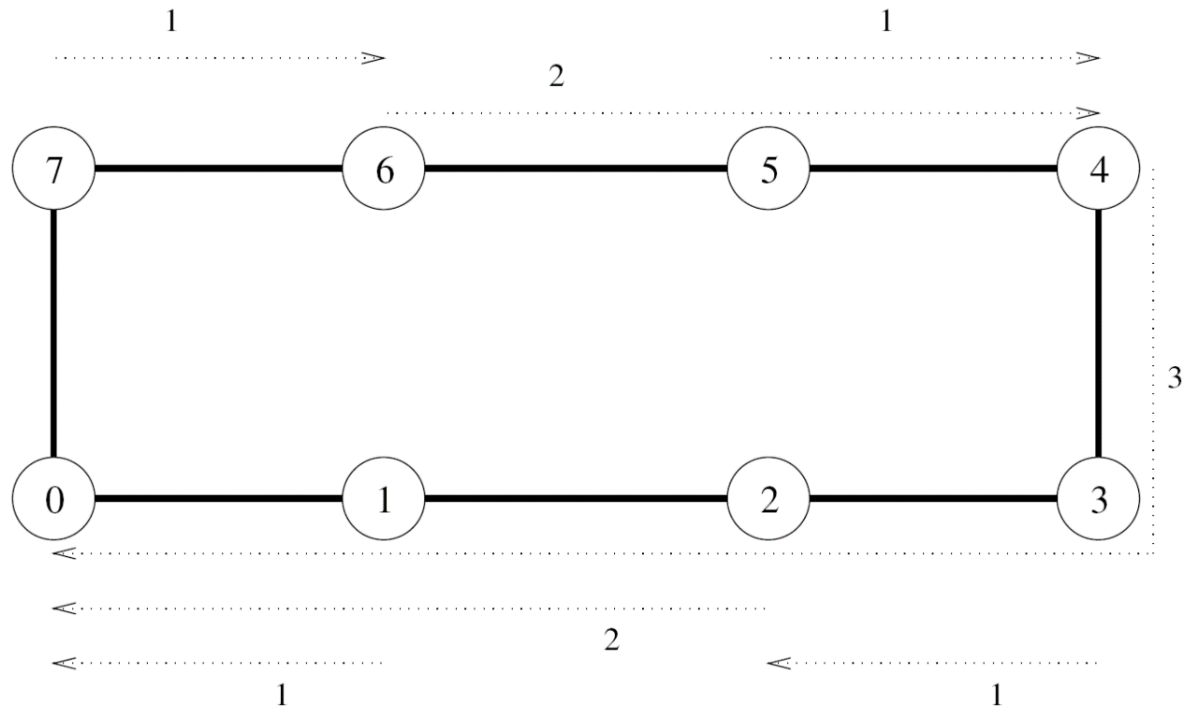


**Figure 4.3** Reduction on an eight-node ring with node 0 as the destination of the reduction.

# Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- Matrix-Vector Multiplication (An Application)

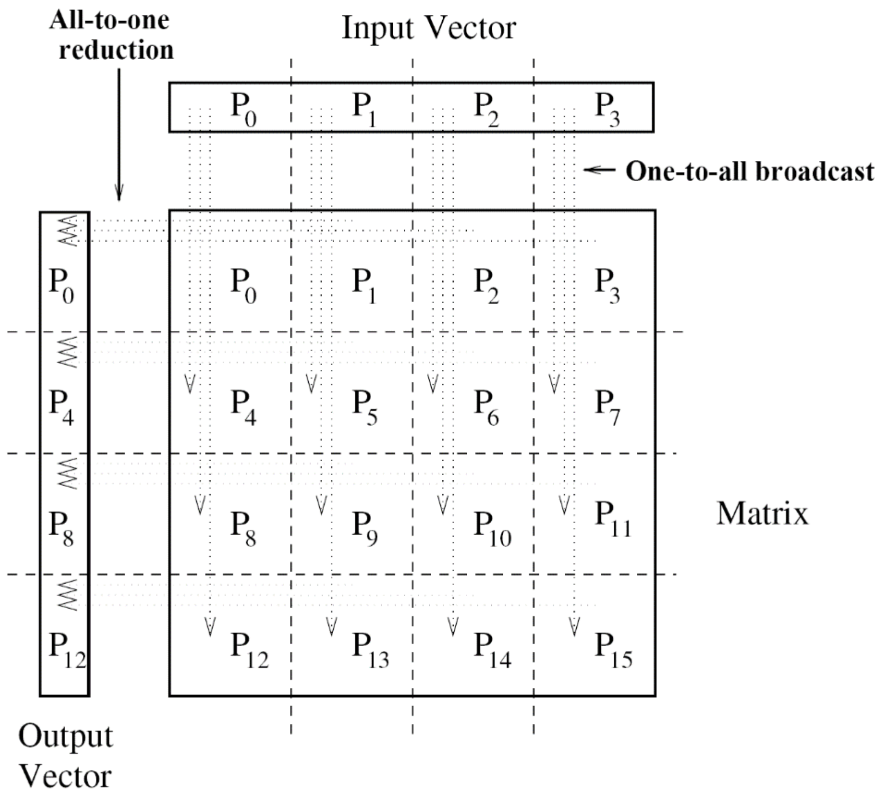


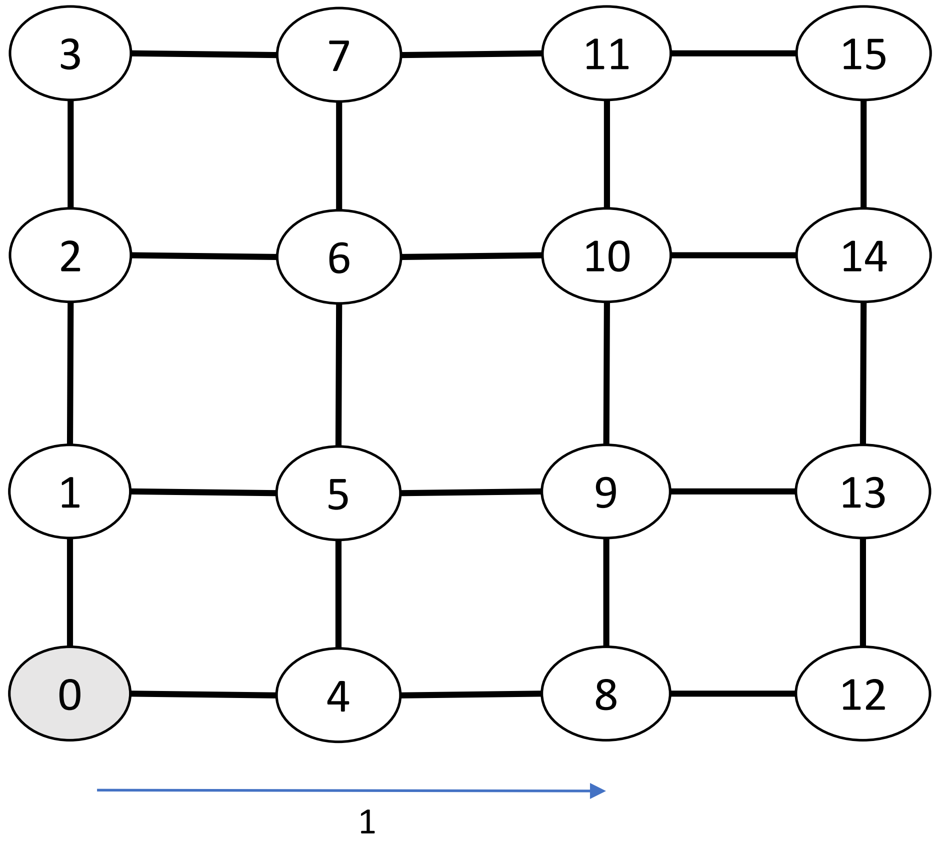
**Figure 4.4** One-to-all broadcast and all-to-one reduction in the multiplication of a $4 \times 4$ matrix with a $4 \times 1$ vector.

# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh**
  - We can regard each row and column of a square mesh of *p* nodes as a linear array of nodes
  - Communication algorithms on the mesh are simple extensions of their linear array counterparts
  - **Broadcast and Reduction**
    - Two step breakdown:
      - The operation is performed along one by treating the row as linear array
      - Then all the columns are treated similarly

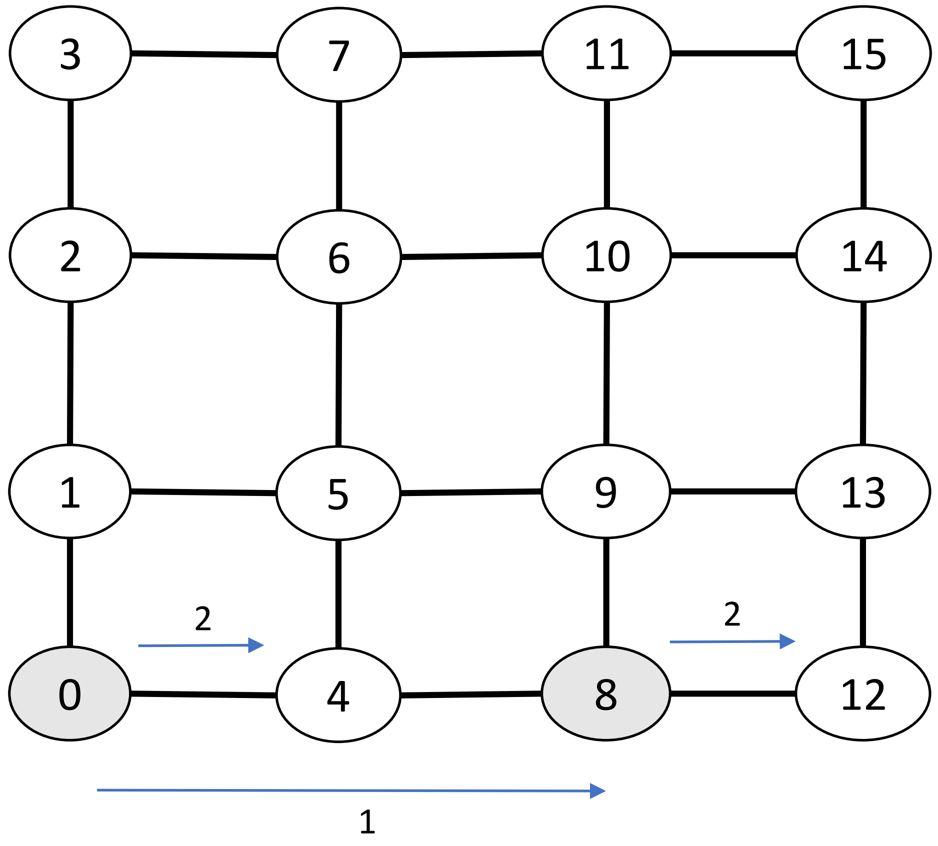
# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



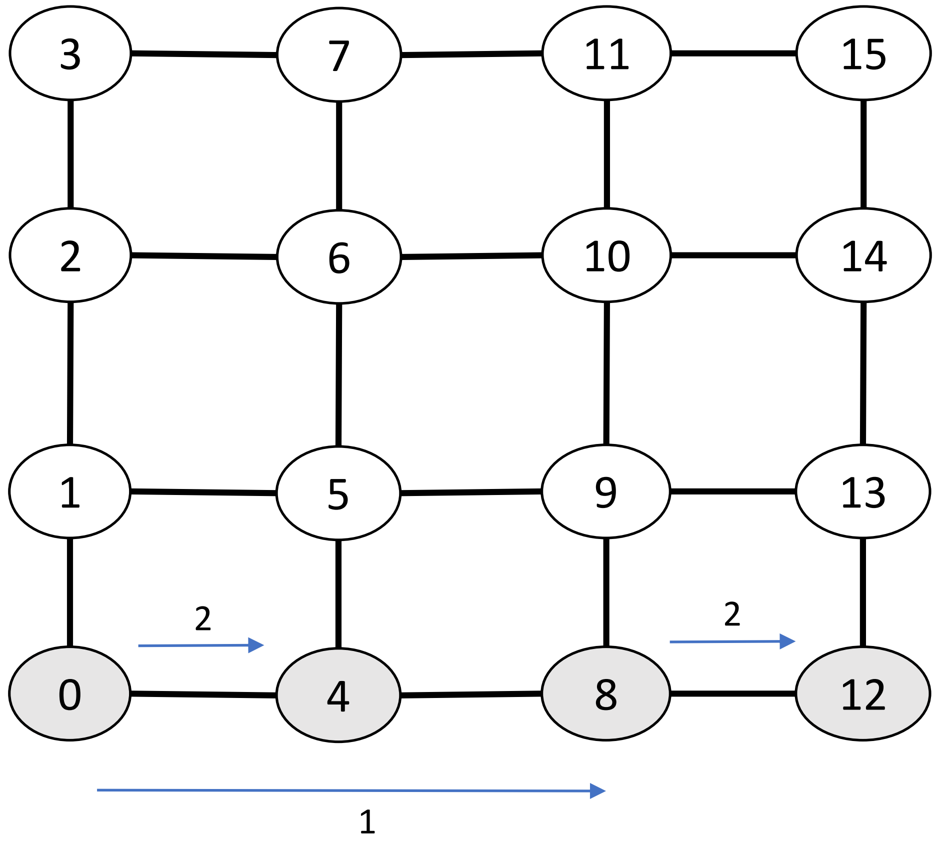
# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



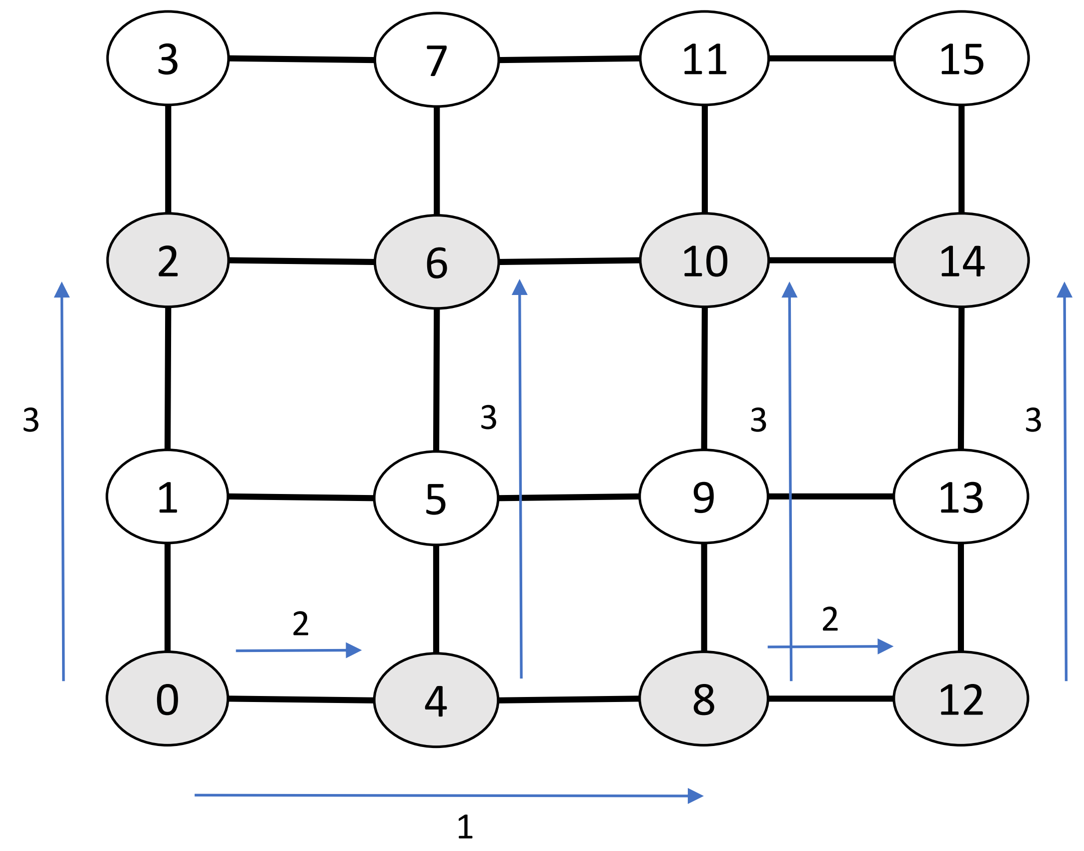
# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



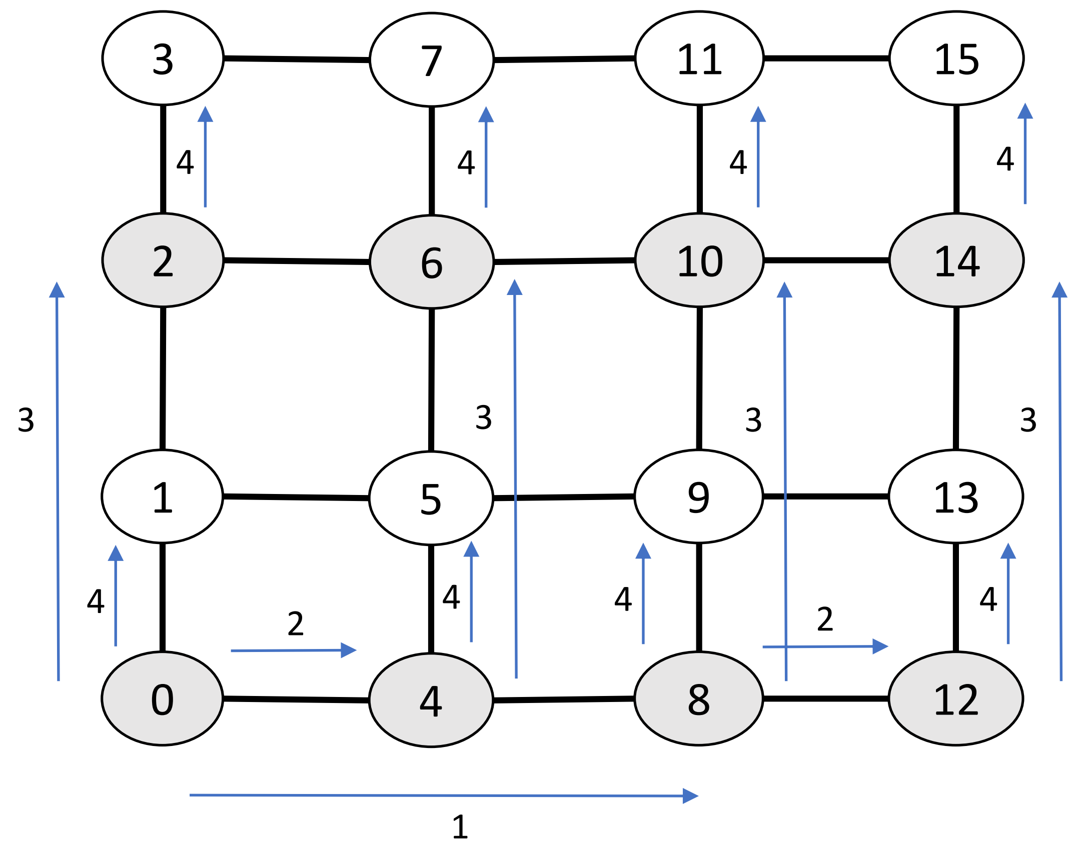
## Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



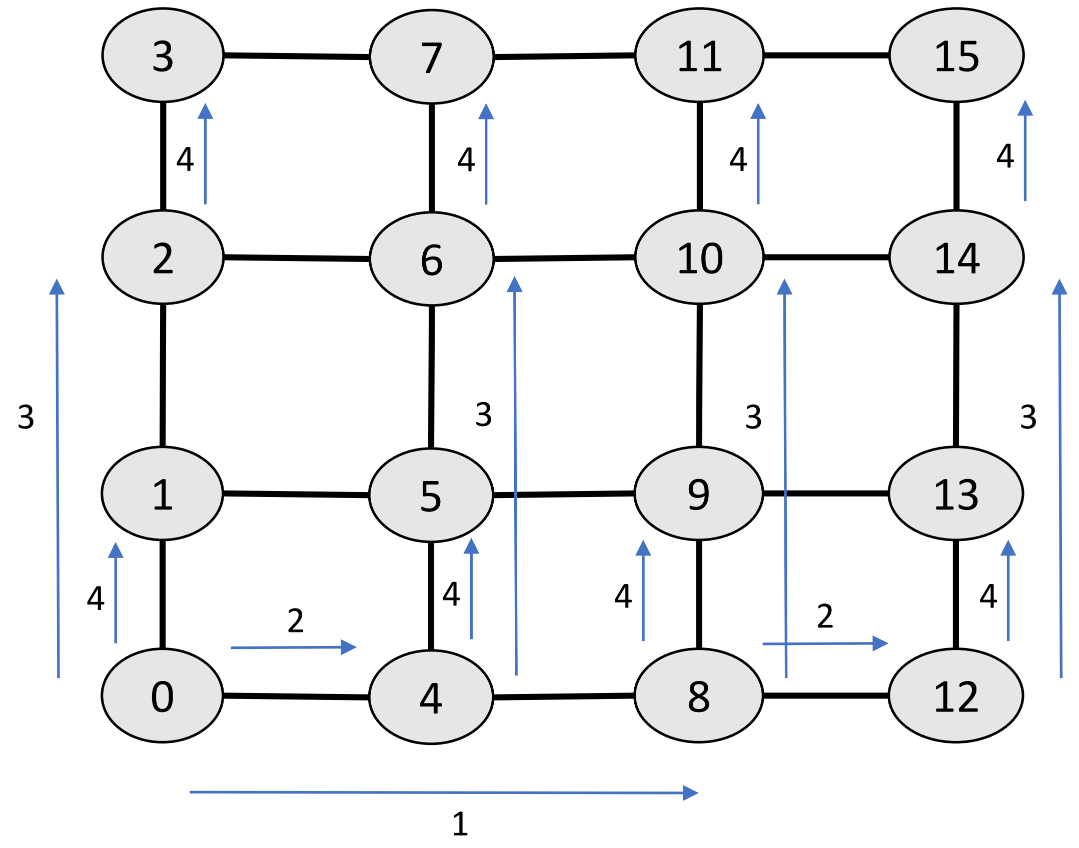
# Basic Communication Operations
(One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**



# Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Mesh (broadcast and reduction)**

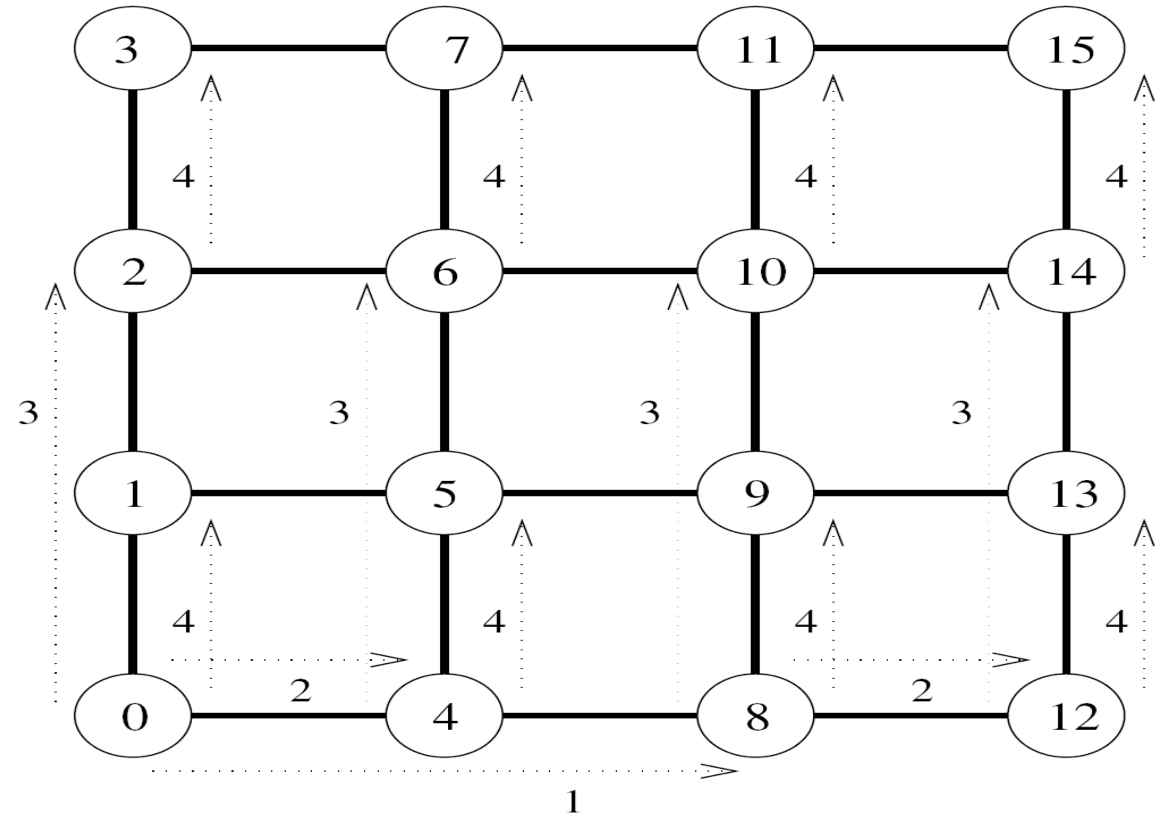


**Figure 4.5** One-to-all broadcast on a 16-node mesh.

## References

1. Slides of Dr. Haroon Mahmood
2. Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). *Introduction to parallel computing* (Vol. 110). Redwood City, CA: Benjamin/Cummings.
3. Quinn, M. J. Parallel Programming in C with MPI and OpenMP, (2003).