

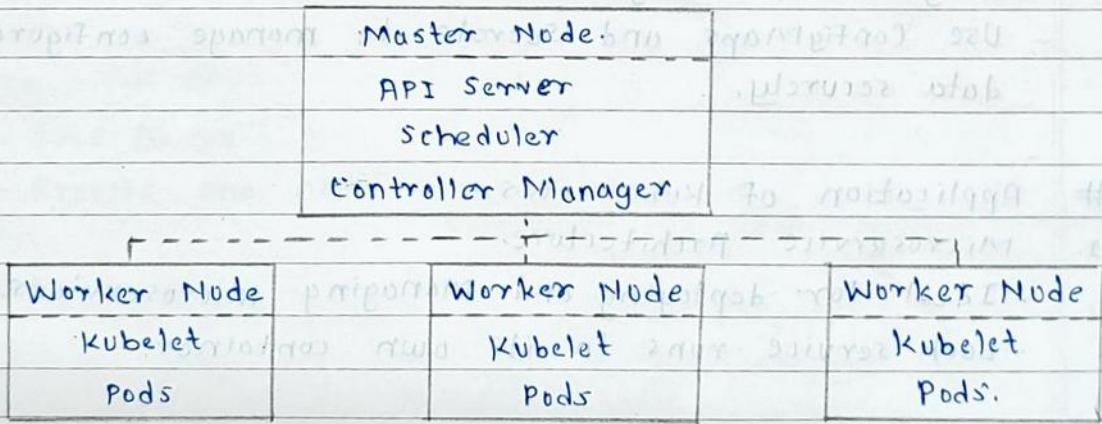
# SPPU-BE-COMP-CONTENT – KSKA Git

## # ASSIGNMENT: NO: 2: (Two):-

Q1> What is Kubernetes? Explain its feature and Applications.

- ANS.
- Kubernetes is an Open-source container orchestration platform originally developed by Google and now maintained by the Cloud Native Computing Foundation.
  - It is used to automate the deployment, scaling, and management of containerized Applications (like those built using Docker).
  - In simple terms, Kubernetes help you run applications in containers across multiple machines efficiently and reliably.

### o Architecture (Block Diagram)



## # Key Features of Kubernetes.

1. Container Orchestration,
  - Manages multiple containers across a cluster of machines.
  - Automatically schedules container on available resources.

# SPPU-BE-COMP-CONTENT - KSKA Git

## 2. Auto Scaling

- Automatically increases or decreases the number of containers (pods) based on load.
- Support Horizontal Pod Scaling (HPA)

## 3. Self Healing

- Restarts failed Containers.
- Replaces unhealthy nodes.
- Ensures desired state is always maintained.

## 4. Load Balancing And Service Discovery.

- Distributes traffic across containers.
- Provides internal DNS for service discovery

## 5. Automated Rollouts and Rollbacks.

- Deploys updates gradually.
- Roll back to previous version if something fails.

## 6. Configuration Management.

- Use ConfigMaps and Secrets to manage configuration data securely.

## # Application of Kubernetes:-

### 1. Microservice Architecture.

- Ideal for deploying and Managing microservices.
- Each service runs in its own container.

### 2. DevOps and CI/CD Pipelines.

- Automates build, test, and deployment processes.
- Integrates with tools like Jenkins, GitHub Actions.

### 3. Cloud Native Applications.



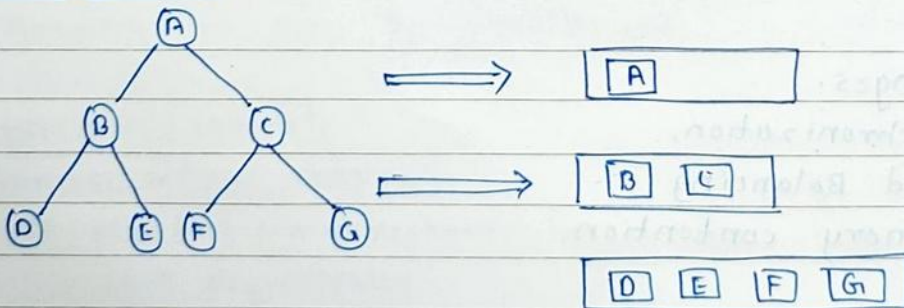
# SPPU-BE-COMP-CONTENT - KSKA Git

- Designed for scalable and resilient cloud Apps.
  - Support modern application development practices.
4. Big Data and AI/ML Workloads.
- Runs Distributed computing Frameworks.
  - Used in AI/ML pipelines and data processing.
5. Web Hosting & SaaS Platforms.
- Host websites and web apps with high availability.
  - Used by companies for SaaS product delivery.
6. Hybrid and Multi-cloud Deployments.
- Enables workload portability across multiple cloud providers.

Q2) Explain BFS for parallel execution and analyze its complexity

Ans: Breadth First Search (BFS)

- It is a traversal technique in which the graph/tree is visited (traversed) level by level using a 'Queue' data structure.
- Sequential BFS:-
  - Uses Queue
  - Process one node at a time.
- BFS Queuing:-



# SPPU-BE-COMP-CONTENT - KSKA Git

## Parallel BFS Concept:-

In parallel BFS:

- All Nodes at the same level are processed simultaneously.
- Uses Frontier based Approach.

## → Parallel BFS Block Diagram.

Level 0 :

A

Level 1 :

B

C

(processed in parallel)

Level 2 :

D

E

(processed in parallel.)

## Algorithm Steps:-

- ① Initialize source node.
- ② Add to frontier (Queue)
- ③ For each node in Frontier (parallel),  
- Visit Neighbours.
- ④ Create next Frontier.
- ⑤ Repeat until empty.

## Complexity Analysis:-

Sequential BFS :-

$$O(V+E)$$

Parallel BFS :-

$$O\left(\frac{V+E}{P}\right) + \text{communication overhead.}$$

Where, P = number of processors.

## # Challenges.

- Synchronization.
- Load Balancing
- Memory contention.



# SPPU-BE-COMP-CONTENT - KSKA Git

Q3> Compare an Algorithm for Sequential and Parallel Merge Sort. Analyze the complexity for the same.

Ans. 1. Sequential Merge Sort

Sequential Merge sort follows Divide and Conquer:

- Divide Array into two halves.
- Recursively sort both halves.
- Merge sorted halves.

Algorithm (Sequential)

MERGE\_SORT(A, left, right):

if left < right:

mid = (left + right) / 2

MERGE\_SORT(A, left, mid)

MERGE\_SORT(A, mid+1, right)

MERGE(A, left, mid, right)

• Time Complexity :-

Recurrence relation :  $T(n) = 2T(\frac{n}{2}) + O(n)$

By Master's Theorem :-

Best, Average, Worst case :  $O(n \log n)$

2. Parallel Merge Sort

Parallel Merge sort improves performance by:

- Sorting subarrays simultaneously on multiple processors.
- Merging can also be parallelized.

• Algorithm (Parallel)

PARALLEL-MERGE\_SORT(A):

if size(A) <= threshold:

sort sequentially

else:

# SPPU-BE-COMP-CONTENT – KSKA Git

divide A into two halves.

parallel:

PARALLEL-MERGE-SORT (left half)

PARALLEL-MERGE-SORT (right half)

PARALLEL-MERGE (left, right)

## Parallel Time Complexity:-

Let  $p$  = number of processors.

• Sorting levels run in parallel.

• Depth of recursion =  $\log n$ .

Parallel Time:  $T_p(n) = O\left(\frac{n \log n}{p} + \log n\right)$

- Cost (Efficiency measure) =  $p \times T_p(n) = O(n \log n)$

- cost optimal Algorithm

Feature	Sequential Merge Sort.	Parallel Merge Sort.
① Execution	Single Processor	Multiple Processors
② Time Complexity	$O(n \log n)$	$O\left(\frac{n \log n}{p} + \log n\right)$
③ Speed	Slower	Faster.
④ Space Complexity	slower for large data	$O(n)$
⑤ Overhead	None	Thread / process overhead.
⑥ Efficiency	Fixed	Improves with $p$ .

## Parallel merge sort is ideal for:

Multi core GPU's, GPU's, Distributed systems.



# SPPU-BE-COMP-CONTENT - KSKA Git

Page No. :  
Date : / /

Q4.) Explain parallel Depth First Search Algorithm in Detail.

ANS.

Parallel DFS is an extension of traditional DFS where:-

- Multiple processors/threads explore different branches of a graph simultaneously.
- Each processor performs DFS on a portion of the search space.

Goal: Reduce travel time for Large Graph.

• Key Idea:

In Sequential DFS:

• One path is explored at a time (using stack/recursion)

In Parallel DFS:

- When multiple neighbors exists  $\rightarrow$  they are explored in parallel
- Work is divided among processors dynamically.

• Working Principle:-

1. Start from the source Node.
2. Assign different neighbors to different processors.
3. Each processor:
  - performs DFS Independently,
  - Maintains its own stack.
4. Use a shared visited array to avoid revisiting nodes.
5. Synchronize access to shared data.

• Algorithm:-

PARALLEL\_DFS(Graph G, Node start):

mark start as visited

push start into stack

parallel workers:

while stack is not empty:

node = pop from stack.

# SPPU-BE-COMP-CONTENT - KSKA Git

for each neighbor of node:

if not visited:

mark visited

push neighbor into stack

## o TYPES OF PARALLEL DFS:-

1. Stack Splitting : Divide DFS stack among processors.
2. Work stealing DFS: Idle processor steal node from others.
3. Recursive parallel DFS: Recursive calls executed in parallel.

## o Parallel DFS Complexity:-

Same as Sequential DFS :  $O(n+m)$

$n$ : number of vertices,  $m$ : number of Edges.

## o Advantages:-

1. Faster traversal for Large Graphs.
2. Utilizes multicore processors.
3. Scalable for Distributed systems.

## o Limitations:-

1. Synchronization overhead
2. Race conditions and Load imbalance.
3. Harder to implement than sequential DFS.

## o Applications:-

1. AI Search (state space exploration)
2. Maze solving
3. Graph connectivity.
4. Web crawling
5. Parallel Backtracking problems.



# SPPU-BE-COMP-CONTENT - KSKA Git

Q5. Write Short Notes On:-

1. GPU Application

ANS.

A GPU (Graphic Processing Unit) is a highly parallel processor designed to handle multiple computations simultaneously. Unlike CPUs, GPUs contain thousands of smaller cores optimized for parallel processing, making them ideal for data-intensive tasks.

# Applications of GPU:-

1. Graphics Rendering,

- 3D Rendering in Games and animation.
- Real time image processing.

2. AI and ML

- Training deep learning models
- Neural Networks (e.g. RNN, CNN)

3. Scientific Computing

- Weather Forecasting.
- Molecular modeling and simulations.

4. Data Analytics and Big Data.

- Parallel Data processing.
- Faster database queries.

5. Cryptocurrency Mining.

- Efficient hashing computations.

6. Video Processing

- Encoding, Decoding and streaming.

7. Autonomous systems.

- Used in self driving cars for Real time decision making

Advantages:-

1. High Parallelism
2. Faster computation
3. Efficient and scalable.



# SPPU-BE-COMP-CONTENT – KSKA Git

## 2. Parallel Merge sort

Ans. Parallel Merge sort is an extension of the traditional Merge sort algorithm that uses multiple processors to sort data simultaneously using the divide and conquer approach.

### o Working Principle:

1. Divide the Array into subarrays.
2. Assign subarrays to different processor/threads.
3. Sort subarrays in parallel.
4. Merge sorted subarrays in parallel.

### o Algorithm (steps)

1. Split array into halves recursively.
2. Each processor sorts its part independently.
3. Merge results in a hierarchical (tree-like) manner.

### o Example:-

