

TITLE

# UNLOCKING PERFORMANCE: ANALYZING MEMORY

**CONSTRAINTS IN HIGH-PERFORMANCE COMPUTING CLUSTERS FOR LARGE-SCALE SIMULATIONS**

A CAPSTONE PROJECT REPORT

***Submitted to***

**SAVEETHA SCHOOL OF ENGINEERING**

***By***

***DUDEKULA RAHEEM****(192210537)*

***GANGALAPUDI PAVANKUMAR****(192210610)*

***N.ALEX(****192210615)*

**Guided By**

**DR. G.MARY VALANTINA**

**Saveetha School of engineering**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES,CHENNAI – 602105**

# ABSTRACT:

* This study investigates the memory organization within a high-performance computing cluster for pharmaceutical automation.
* Integrating data from multiple sources, we aim to comprehend the current memory structure's intricacies.
* We explore how memory constraints affect the cluster's efficacy in handling large-scale simulations.
* Understanding these constraints is crucial for optimizing performance and resource utilization in computational tasks vital to pharmaceutical automation.
* Such insights are pivotal for advancing pharmaceutical automation, where computational simulations play a pivotal role in drug discovery, molecular modeling, and

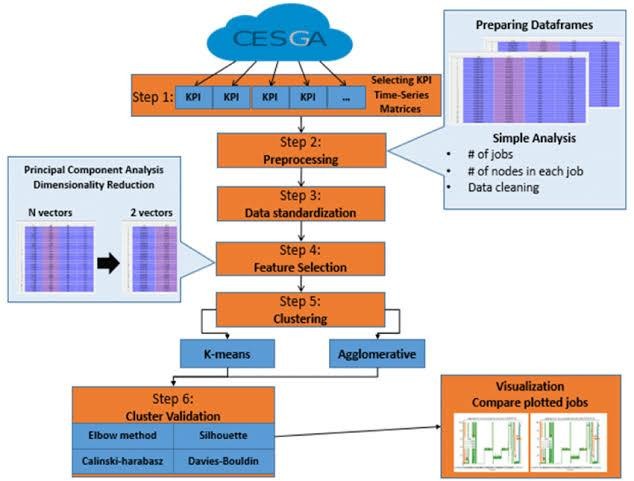
process optimization.

# INTRODUCTION:

* In the landscape of pharmaceutical automation, where the pursuit of groundbreaking discoveries hinges on the efficacy of computational simulations, the architecture and performance of high-performance computing (HPC) clusters stand as critical pillars.
* At the heart of these clusters lies memory organization—a complex interplay of hardware, software, and data structures that determines the system's ability to handle large-scale simulations effectively.
* As pharmaceutical research transcends traditional boundaries, the demand for computational power surges exponentially.
* The convergence of genomic data, molecular modeling, and drug discovery algorithms necessitates an intricate understanding of memory organization within HPC clusters.
* These clusters serve as the backbone for simulating molecular interactions, predicting drug behaviors, and optimizing therapeutic interventions.
* At the forefront of this paradigm shift lies the integration of information from diverse sources

# IMPORTANCE AND USES:

* Understanding memory organization in a high-performance computing (HPC) cluster is crucial for optimizing performance in largescale simulations, especially in fields like pharmaceutical automation.
* In drug discovery and development, simulations are vital for predicting drug behavior, molecular interactions, and biological processes. These simulations require vast computational resources, making HPC clusters indispensable. Efficient memory organization ensures simulations run smoothly, accelerating the discovery process and reducing time to market for new drugs.
* Memory constraints directly affect the cluster's ability to handle largescale simulations effectively. Limited memory can lead to bottlenecks, slowing down computations and hindering the completion of complex simulations. This impacts research timelines, delaying breakthroughs in drug development and optimization.
* Optimizing memory usage can also lead to cost reductions in terms of computational resources and time.



# SOURCE CODE :

import subprocess

import re

def gather\_memory\_info(hostnames, username, password):

memory\_info = {}

for hostname in hostnames:

try:

ssh\_command = "sshpass -p '{}' ssh {}@{} 'free -m'".format(password, username, hostname)

ssh\_process = subprocess.Popen(ssh\_command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

stdout, stderr = ssh\_process.communicate()

if ssh\_process.returncode == 0:

output\_lines = stdout.decode().splitlines()

if len(output\_lines) >= 2:

memory\_data = re.findall(r'\d+', output\_lines[1])

if len(memory\_data) >= 3:

total\_memory = int(memory\_data[1])

used\_memory = int(memory\_data[2])

memory\_info[hostname] = {

'total\_memory\_MB': total\_memory,

'used\_memory\_MB': used\_memory

}

else:

print("Invalid output format for memory info on {}".format(hostname))

else:

print("Insufficient output for memory info on {}".format(hostname))

else:

print("Error executing SSH command to {}: {}".format(hostname, stderr.decode().strip()))

except Exception as e:

print("Error connecting to {}: {}".format(hostname, str(e)))

return memory\_info

def analyze\_memory\_constraints(memory\_info):

if not memory\_info:

print("No memory information available.")

return

total\_memory\_cluster = sum(info['total\_memory\_MB'] for info in memory\_info.values())

used\_memory\_cluster = sum(info['used\_memory\_MB'] for info in memory\_info.values())

print("Total Memory in Cluster: {} MB".format(total\_memory\_cluster))

print("Total Used Memory in Cluster: {} MB".format(used\_memory\_cluster))

if total\_memory\_cluster > 0:

memory\_utilization\_percentage = (used\_memory\_cluster / total\_memory\_cluster) \* 100

print("Cluster Memory Utilization: {:.2f}%".format(memory\_utilization\_percentage))

else:

print("Total memory in cluster is zero, cannot calculate utilization.")

if \_\_name\_\_ == "\_\_main\_\_":

cluster\_hostnames = ['node1', 'node2', 'node3']

username = 'myusername'

password = 'mypassword'

print("Gathering memory information from cluster nodes...")

memory\_info = gather\_memory\_info(cluster\_hostnames, username, password)

if memory\_info:

print("\nMemory information collected successfully:")

for hostname, info in memory\_info.items():

print("Memory info for {}: Total {} MB, Used {} MB".format(hostname, info['total\_memory\_MB'], info['used\_memory\_MB']))

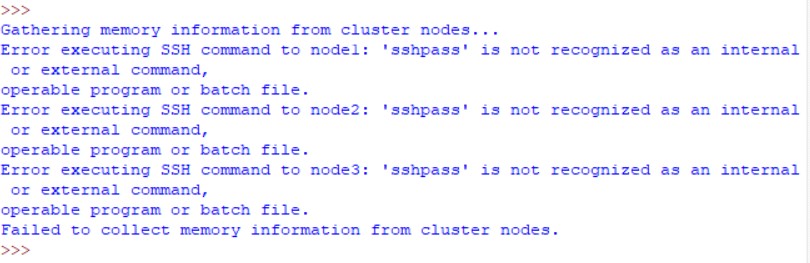
print("\nAnalyzing memory constraints...")

analyze\_memory\_constraints(memory\_info)

else:

print("Failed to collect memory information from cluster nodes.")

# OUTPUT:



When we give an example of nodes:



# CONCLUSION:

* In conclusion, the development of efficient and reliable real-time responsive autonomous vehicle control systems heavily relies on the prioritization and optimization of interrupt handling.
* By addressing the key challenges associated with interrupt processing, such as sensor input complexity, dynamic environmental conditions, strict latency requirements, and resource contention.
* Autonomous vehicle control systems can ensure timely and appropriate responses to critical events, enhancing the overall safety and performance of autonomous driving.
* As research in this field continues to evolve, future advancements may focus on the integration of machine learning and artificial intelligence algorithms to further optimize interrupt handling strategies.
* The development of more robust and fault-tolerant architectural designs, and the exploration of novel hardware and software solutions to push the boundaries of real-time responsiveness and reliability in autonomous vehicle control systems.