No. of data centres: There is 3 data center used in this simulation.

No. of hosts: The data center contains 2 hosts (physical machines).

No. of virtual machines: 10.

No. of CPUs: Each host has 1 CPU.

MIPS of CPU per virtual machine (Millions of Instructions Per Second):

Homogenous: Each VM has a MIPS value of 1200.

Heterogeneous: VMs have varying MIPS values greater than 2000 but less than or equal to 20000.

Low performance mips= 1000;

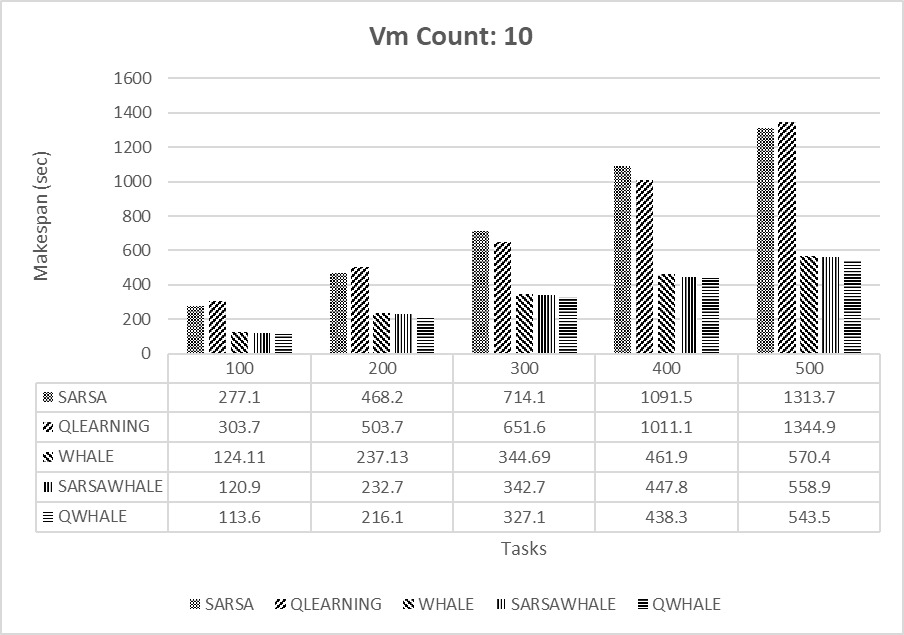
Medium performance mips= 2000;

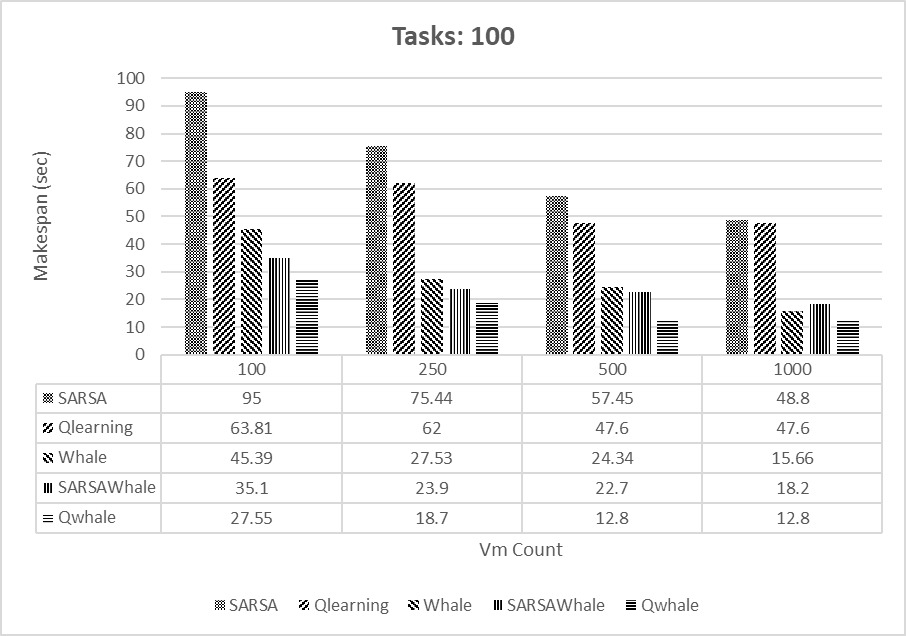
High performance mips = 4000;

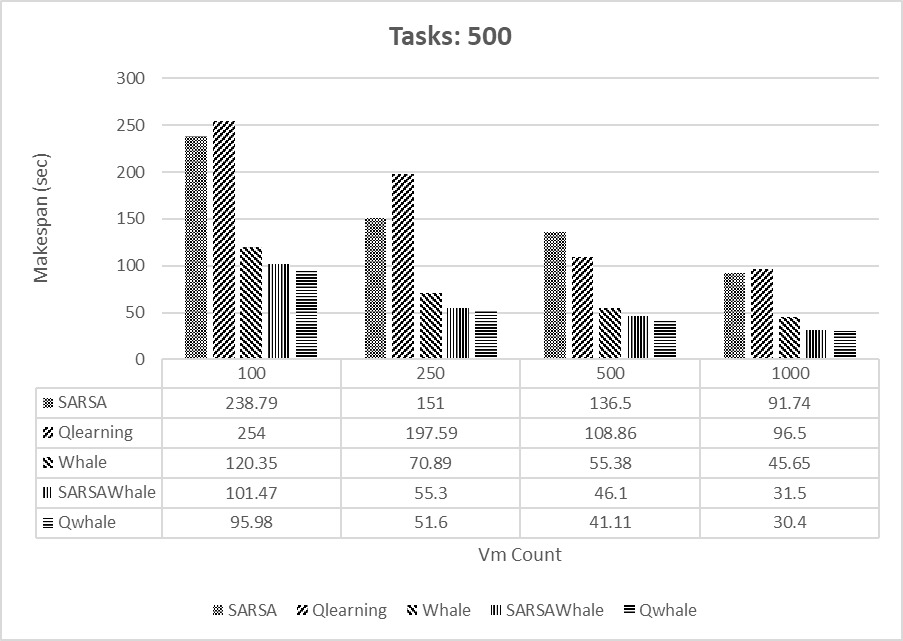
**mips = MIPS \* VM Number;**

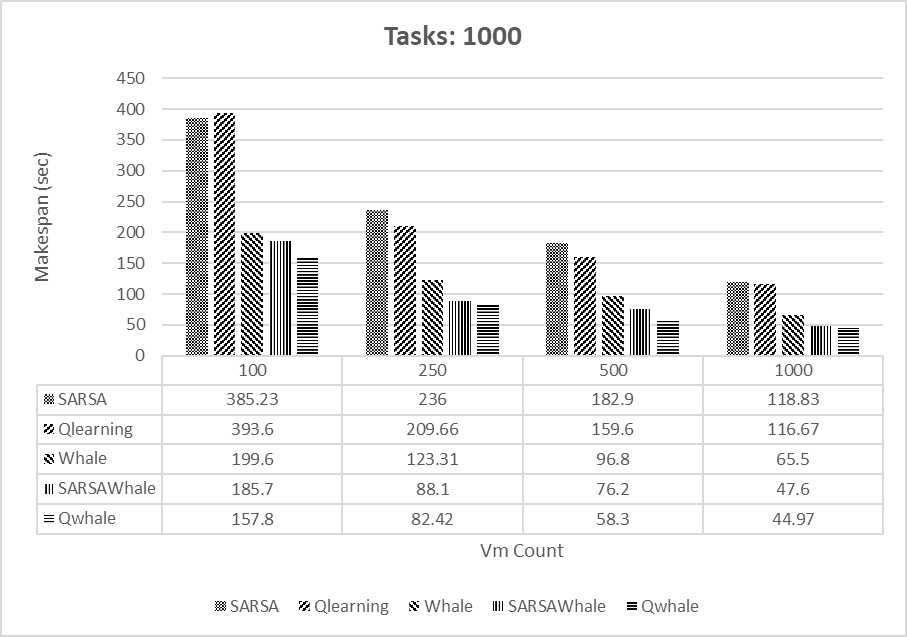
RAM: Each VM is allocated 2 GB of RAM.

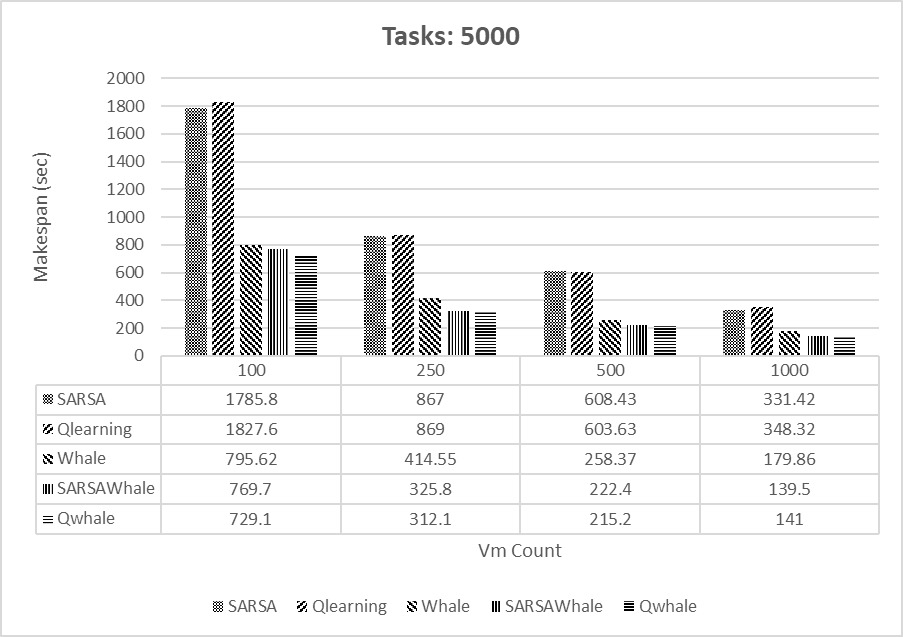
These parameters define the computational resources and network characteristics of the simulated environment, influencing the performance and behavior of the virtual machines during the simulation. The variability in the number of VMs and their heterogeneity in terms of RAM and MIPS allows for a range of scenarios to be tested, which can help in understanding the performance and scalability of the data center under different conditions.











The two charts illustrate the makespan (execution time) in seconds for different task scheduling algorithms under varying numbers of virtual machines (VMs). Each chart represents different workloads: 5000 tasks and 1000 tasks. The algorithms compared are SARSA, Q-learning, Whale, SARSA-Whale, and Q-Whale. Here's a detailed analysis of the results:

Analysis for 5000 Tasks

100 VMs:

SARSA: 1785.8 seconds

Q-learning: 1827.6 seconds

Whale: 795.62 seconds

SARSA-Whale: 769.7 seconds

Q-Whale: 729.1 seconds

Observation: Q-Whale performs the best with the shortest makespan, followed closely by SARSA-Whale and Whale. SARSA and Q-learning have much higher makespans.

250 VMs:

SARSA: 867 seconds

Q-learning: 869 seconds

Whale: 414.55 seconds

SARSA-Whale: 325.8 seconds

Q-Whale: 312.1 seconds

Observation: Q-Whale and SARSA-Whale again outperform the other algorithms, with Q-Whale being slightly better.

500 VMs:

SARSA: 608.43 seconds

Q-learning: 603.63 seconds

Whale: 258.37 seconds

SARSA-Whale: 222.4 seconds

Q-Whale: 215.2 seconds

Observation: Q-Whale and SARSA-Whale maintain their superior performance, with the Whale algorithm still performing significantly better than SARSA and Q-learning.

1000 VMs:

SARSA: 331.42 seconds

Q-learning: 348.32 seconds

Whale: 179.86 seconds

SARSA-Whale: 139.5 seconds

Q-Whale: 141 seconds

Observation: SARSA-Whale achieves the best performance, closely followed by Q-Whale. Whale also shows good performance compared to SARSA and Q-learning.

Analysis for 1000 Tasks

100 VMs:

SARSA: 385.23 seconds

Q-learning: 393.6 seconds

Whale: 199.6 seconds

SARSA-Whale: 185.7 seconds

Q-Whale: 157.8 seconds

Observation: Q-Whale outperforms all other algorithms, with SARSA-Whale being the second-best. Whale is significantly better than SARSA and Q-learning.

250 VMs:

SARSA: 236 seconds

Q-learning: 209.66 seconds

Whale: 123.31 seconds

SARSA-Whale: 88.1 seconds

Q-Whale: 82.42 seconds

Observation: Q-Whale and SARSA-Whale again show the best performance, with Q-Whale slightly ahead.

500 VMs:

SARSA: 182.9 seconds

Q-learning: 159.6 seconds

Whale: 96.8 seconds

SARSA-Whale: 76.2 seconds

Q-Whale: 58.3 seconds

Observation: Q-Whale and SARSA-Whale continue to outperform the other algorithms, with Q-Whale leading.

1000 VMs:

SARSA: 118.83 seconds

Q-learning: 116.67 seconds

Whale: 65.5 seconds

SARSA-Whale: 47.6 seconds

Q-Whale: 44.97 seconds

Observation: Q-Whale achieves the best performance, closely followed by SARSA-Whale. Whale also performs well compared to SARSA and Q-learning.

Overall Observations

Q-Whale consistently shows the best performance across different VM counts and task numbers.

SARSA-Whale is the second-best performer, consistently close to Q-Whale.

Whale algorithm also performs well, better than SARSA and Q-learning.

SARSA and Q-learning generally have higher makespans compared to the other algorithms, indicating less efficiency.

These results indicate that combining Whale optimization with reinforcement learning techniques like Q-learning and SARSA (resulting in Q-Whale and SARSA-Whale) significantly improves scheduling performance in terms of makespan.

**Energy Consumption**

Based on the provided utilization levels, you can categorize the virtual machines' (VMs) CPU utilization into three different scenarios: high, medium, and low utilization. Here is a breakdown of these scenarios:

High Utilization:

CPU Utilization Levels: 80%, 90%, and 100%

This scenario represents the VMs operating under heavy load conditions, where they are using a substantial portion of their available CPU resources.

Medium Utilization:

CPU Utilization Levels: 60%, 70%, and 80%

In this scenario, the VMs are moderately loaded, utilizing a fair amount of their CPU resources but not to their maximum capacity.

Low Utilization:

CPU Utilization Levels: 30%, 40%, and 50%

This scenario depicts the VMs under light load conditions, using only a small fraction of their available CPU resources.

Summary Table with Utilization Levels

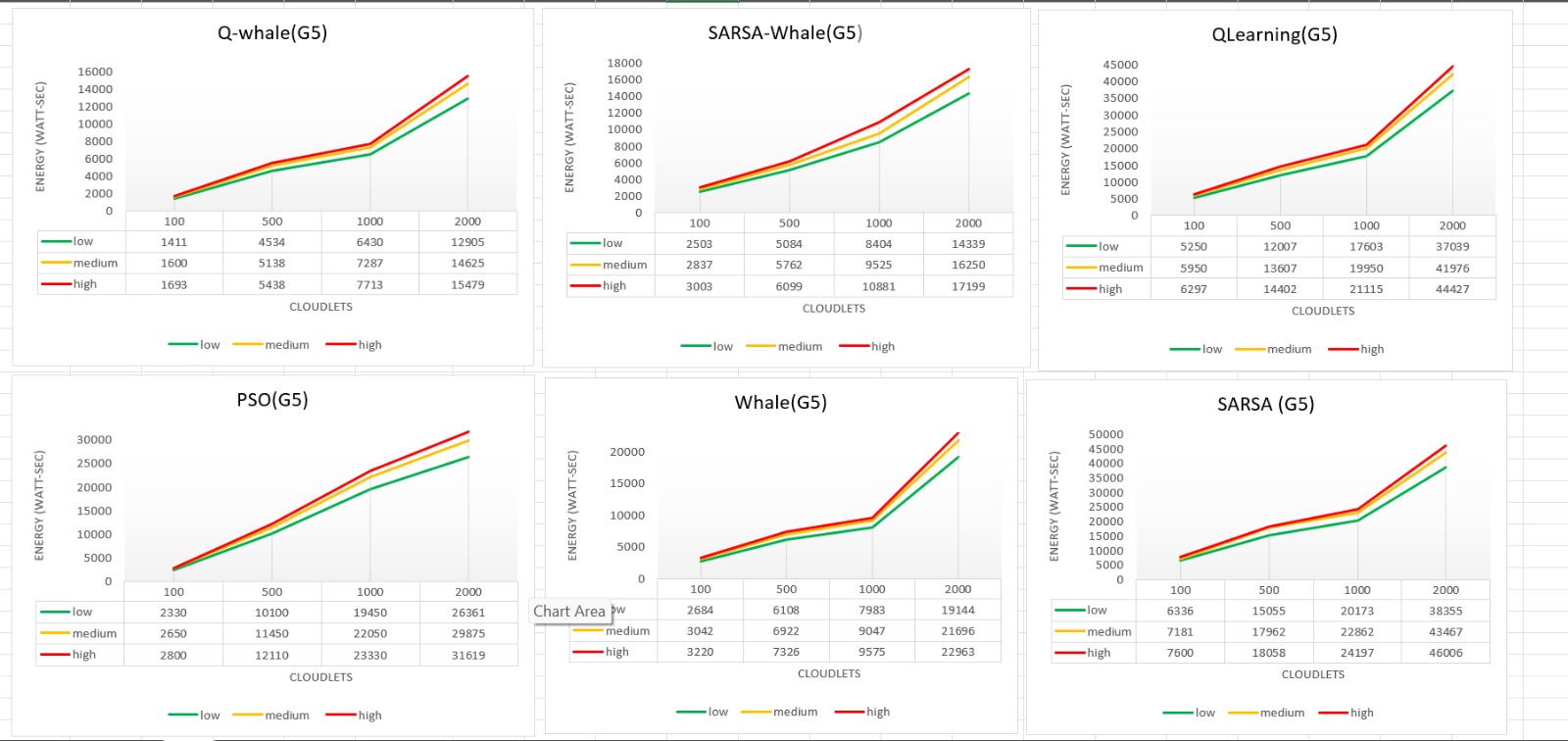
Utilization Category CPU Utilization Levels (%)

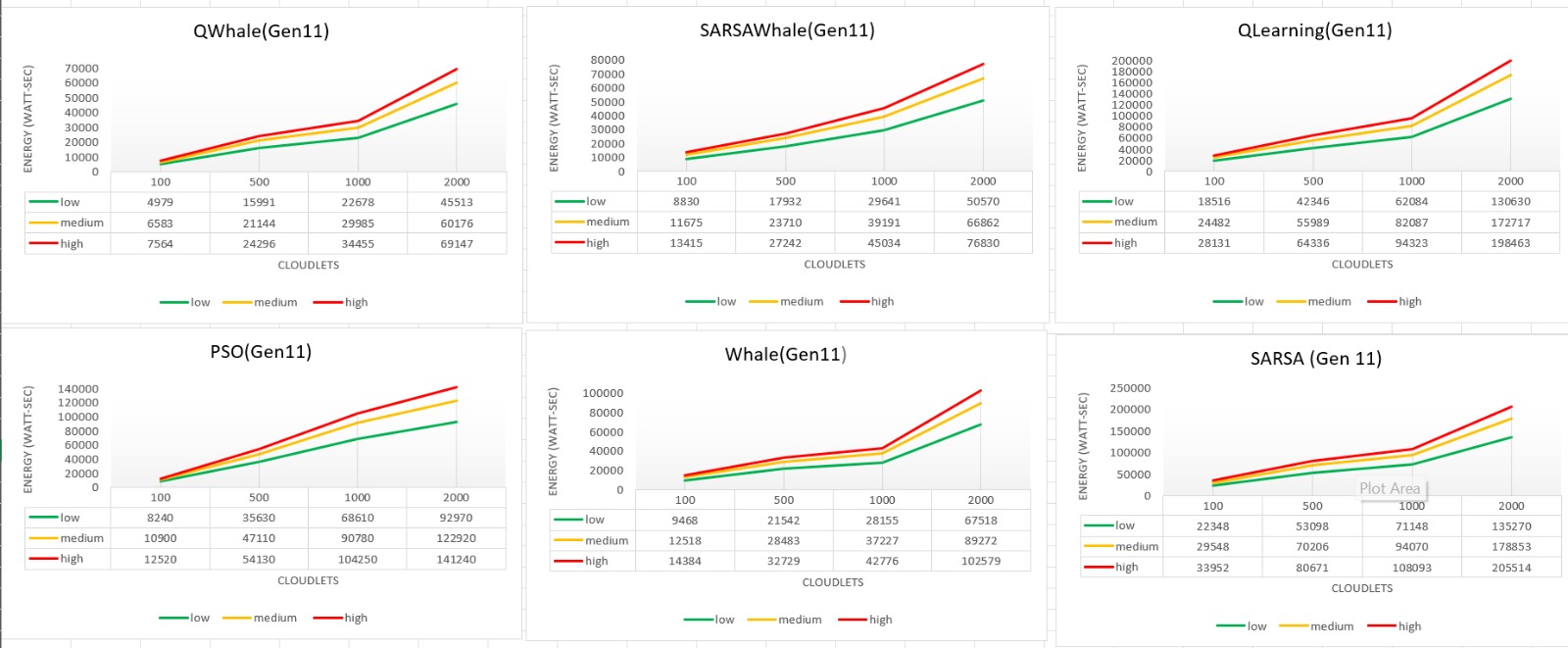
High Utilization 80, 90, 100

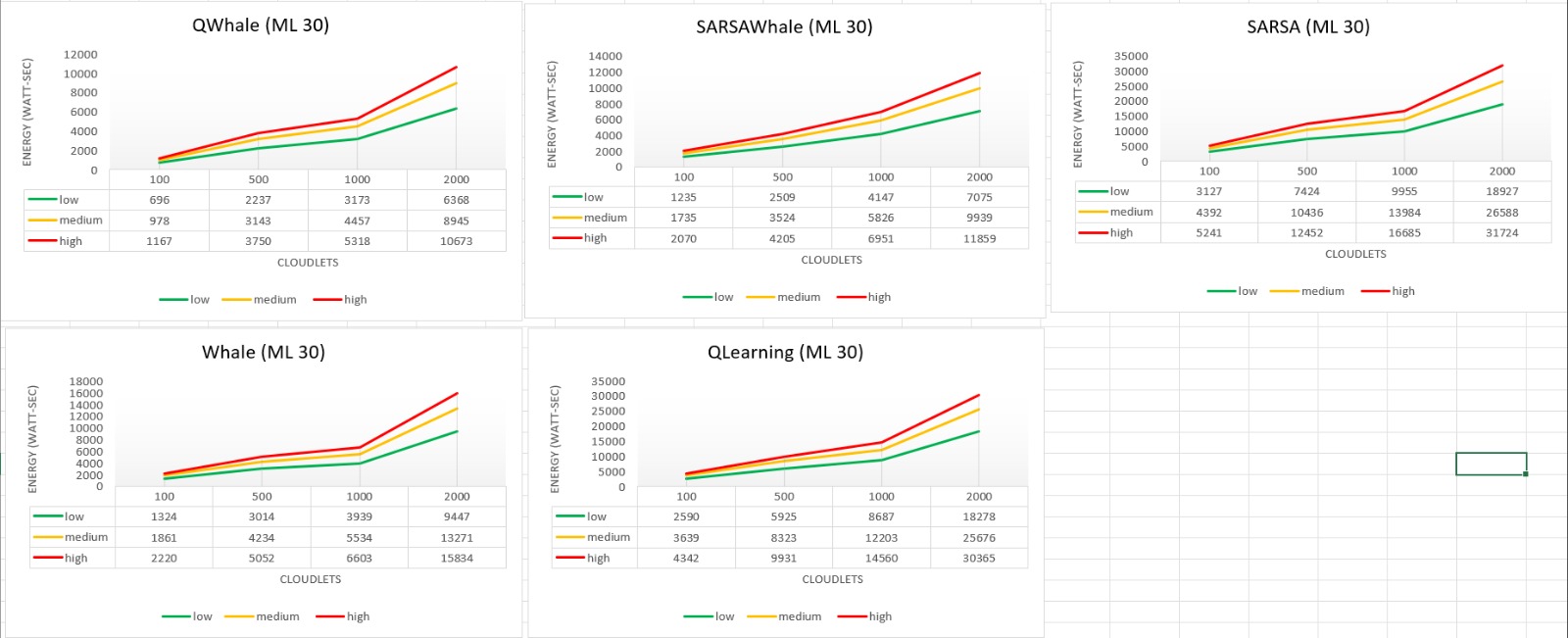
Medium Utilization 60, 70, 80

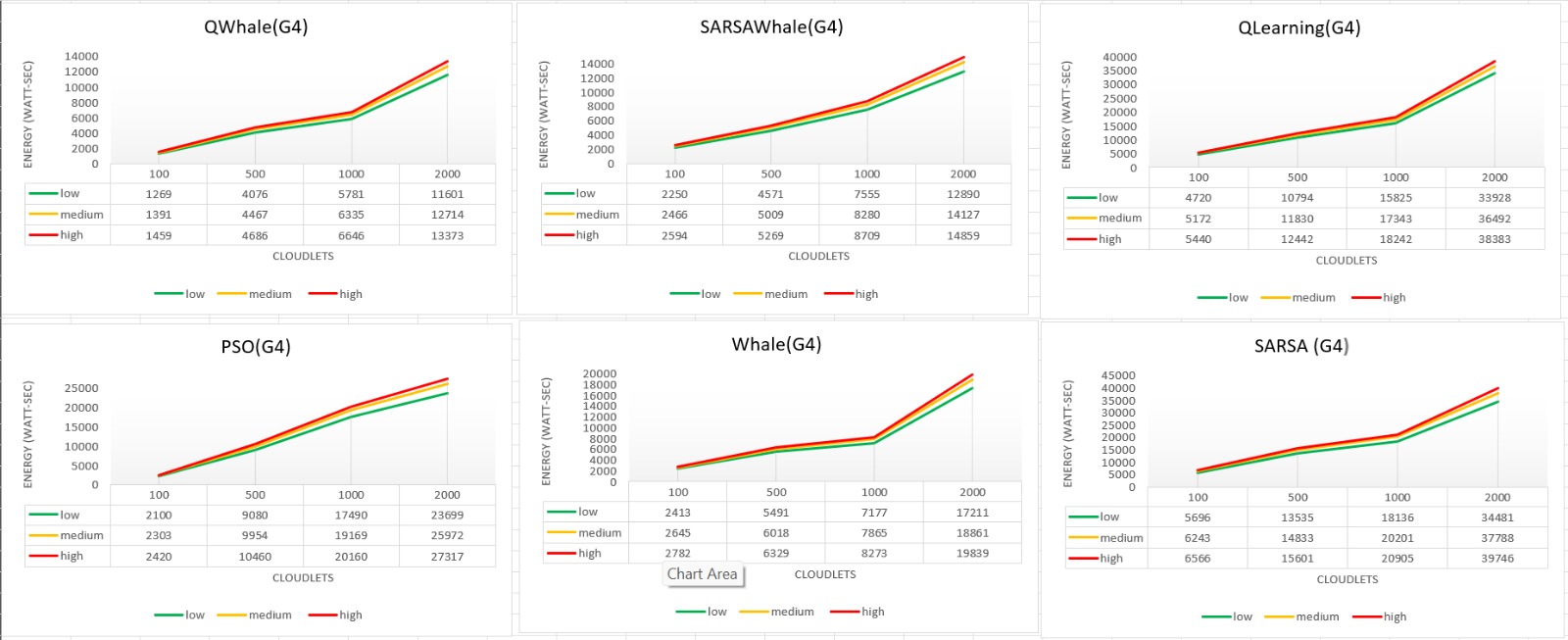
Low Utilization 30, 40, 50

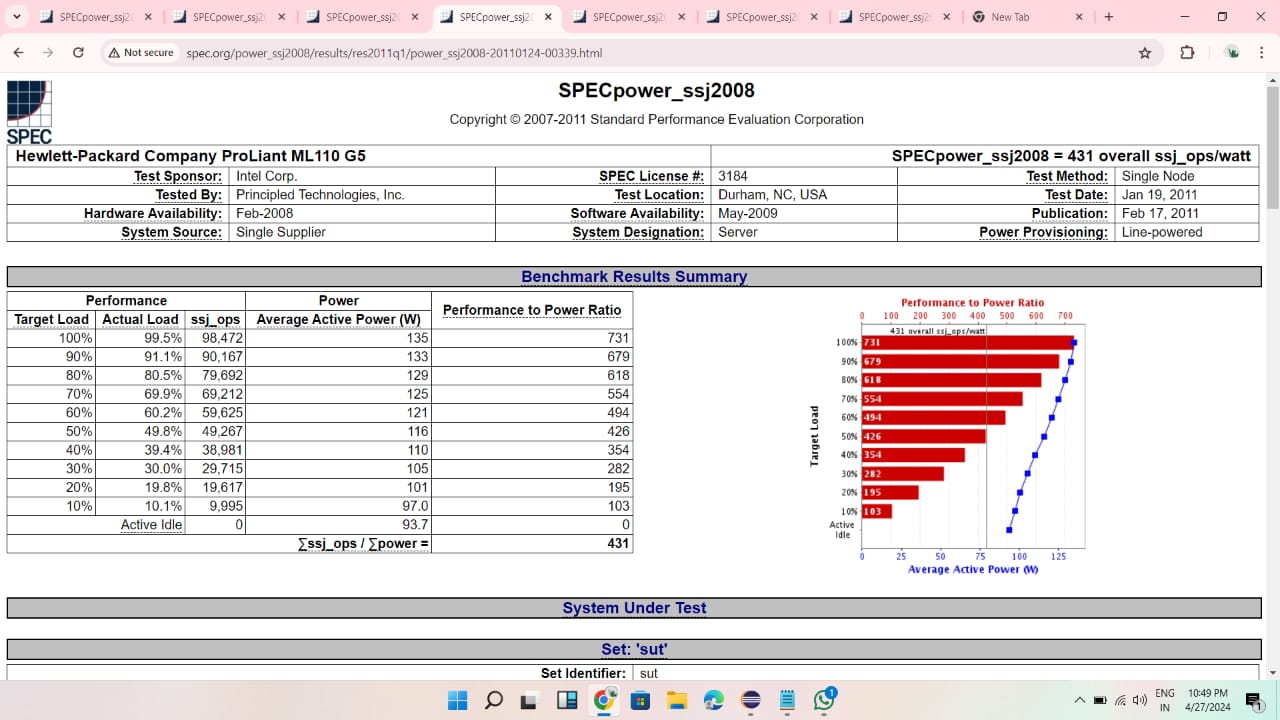
These utilization levels can be used in the simulation to assess how different workloads impact the performance of the data center. By running simulations under these three scenarios, you can observe how the system behaves under various load conditions, which can help in understanding the performance, resource allocation, and potential bottlenecks within the data center.

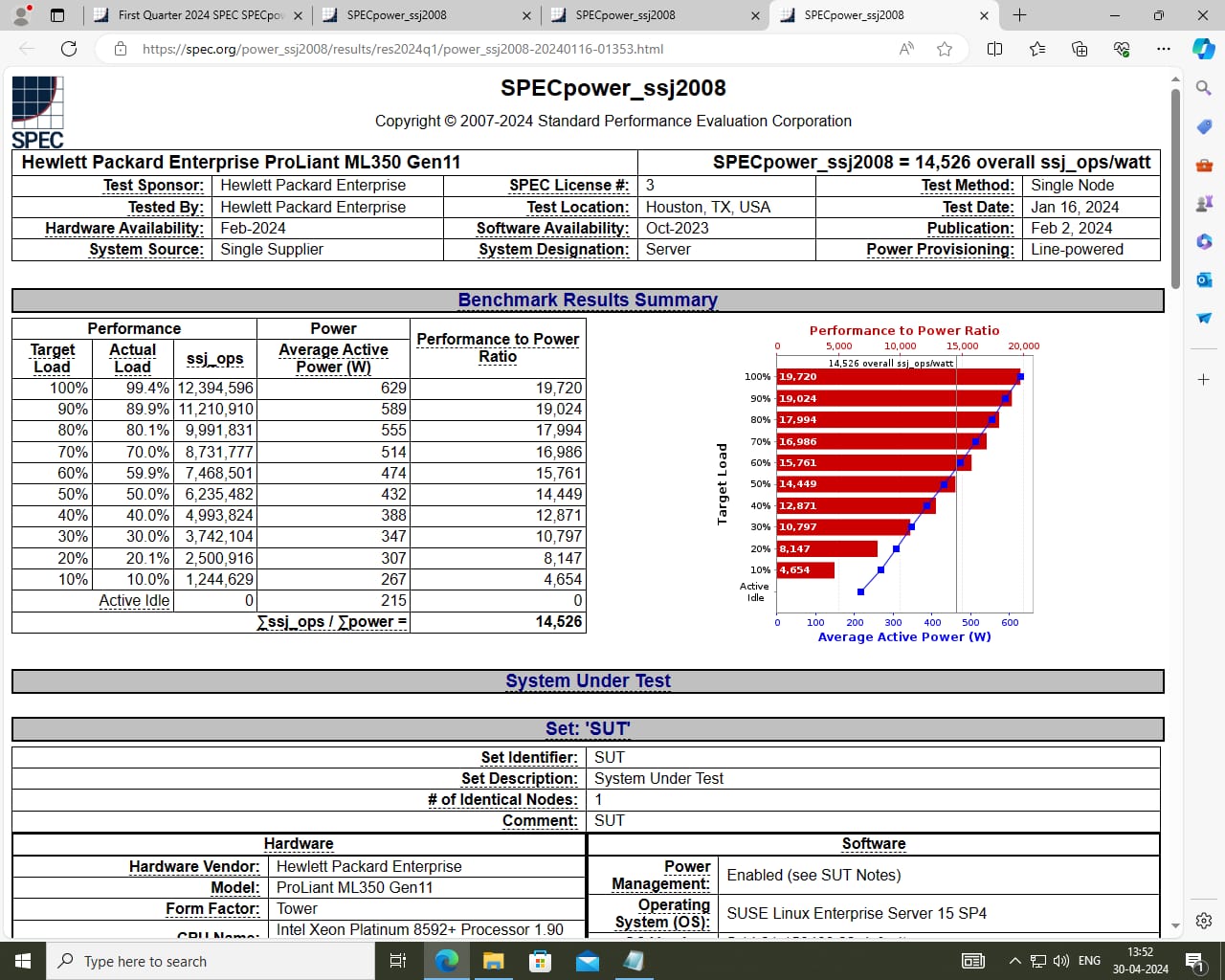
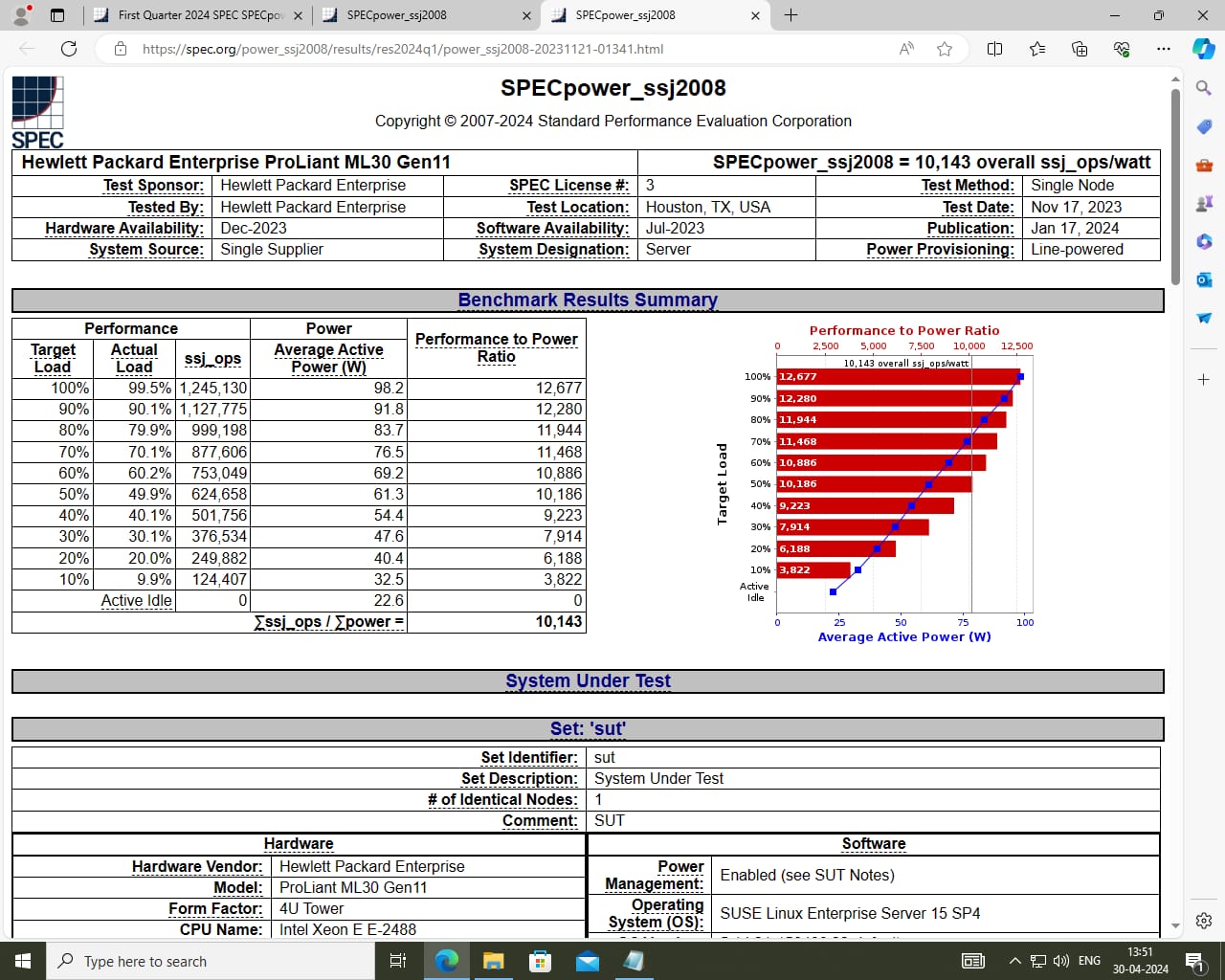












**Energy Consumption=Power(watt)\*Makespan(second)**

**Power-> Overall on the basis of utilization**

**Summary of Data**

* **Energy Consumption at 100 Tasks**:
  + Qwhale: 947
  + SARSAWhale: 1680
  + Qlearning: 3523
  + Whale: 1801
  + SARSA: 4253
* **Energy Consumption at 500 Tasks**:
  + Qwhale: 3043
  + SARSAWhale: 3412
  + Qlearning: 8059
  + Whale: 4100
  + SARSA: 10104
* **Energy Consumption at 1000 Tasks**:
  + Qwhale: 4316
  + SARSAWhale: 5641
  + Qlearning: 11816
  + Whale: 5358
  + SARSA: 13541
* **Energy Consumption at 2000 Tasks**:
  + Qwhale: 8662
  + SARSAWhale: 9624
  + Qlearning: 24773
  + Whale: 12850
  + SARSA: 25746

**Analysis**

1. **Energy Efficiency**:
   * **Qwhale** consistently shows the lowest energy consumption across all tasks, indicating it is the most energy-efficient algorithm.
   * **SARSA** shows the highest energy consumption, making it the least energy-efficient algorithm among those compared.
2. **Trends with Increasing Tasks**:
   * All algorithms show an increasing trend in energy consumption as the number of tasks increases.
   * The rate of increase varies, with Qlearning and SARSA showing more significant increases compared to the others.
3. **Comparative Performance**:
   * **Whale and SARSAWhale** show intermediate energy consumption levels, with Whale generally being slightly more energy-efficient than SARSAWhale.
   * **Qlearning** and **SARSA** show a steep increase in energy consumption, especially noticeable at 1000 and 2000 tasks.

**Observations**

* **Qwhale**: The most energy-efficient, making it suitable for scenarios where energy consumption is a critical factor.
* **SARSA**: The least energy-efficient, which might be suitable for scenarios where energy consumption is less of a concern but perhaps other factors like performance are prioritized.
* **Qlearning**: While it is less energy-efficient, it might offer benefits in terms of learning efficiency or accuracy, which would need further analysis.

Overall, the chart provides a clear comparison of energy consumption trends among different algorithms, highlighting the trade-offs between energy efficiency and the potential need for evaluating other performance metrics.

Energy=Power\*Makespan

**Detailed Analysis**

To delve deeper into understanding the performance of each algorithm, let's interpret how power and makespan contribute to the energy consumption:

1. **Power**: This refers to the average power consumption of the algorithm during its execution.
2. **Makespan**: This refers to the total time taken by the algorithm to complete the tasks.

Given this, an algorithm with lower energy consumption is either more power-efficient, has a shorter makespan, or both.

**Evaluation Based on Energy Formula**

**Qwhale:**

* **Energy Consumption**: The lowest across all task levels.
* **Inference**: Indicates high power efficiency and/or low makespan, making it ideal for energy-constrained environments.

**SARSAWhale:**

* **Energy Consumption**: Moderately low, higher than Qwhale but lower than Qlearning and SARSA.
* **Inference**: Balanced performance, suggesting decent power efficiency and makespan.

**Qlearning:**

* **Energy Consumption**: High, with a steep increase as tasks increase.
* **Inference**: Likely due to higher power consumption and/or longer makespan. This algorithm might prioritize other aspects like accuracy or learning efficiency over energy consumption.

**Whale:**

* **Energy Consumption**: Intermediate, better than SARSA and Qlearning but not as efficient as Qwhale.
* **Inference**: Suggests a balance, with reasonable power consumption and makespan.

**SARSA:**

* **Energy Consumption**: The highest, especially noticeable at higher task levels.
* **Inference**: Indicates high power consumption and/or long makespan, making it less suitable for energy-efficient scenarios.

**Recommendations**

* **Qwhale**: Best suited for scenarios where minimizing energy consumption is crucial.
* **SARSA**: Might be suitable in scenarios where other performance metrics (accuracy, robustness, etc.) are more critical than energy efficiency.
* **Qlearning**: If energy consumption is not a primary concern, Qlearning might offer benefits in learning performance or other metrics.
* **Whale and SARSAWhale**: Provide a middle ground, offering a trade-off between energy efficiency and other potential benefits.

**Conclusion**

The energy consumption chart, along with the energy formula, provides insights into the efficiency of each algorithm. For applications where energy efficiency is paramount, Qwhale stands out as the best choice. However, depending on the specific requirements and constraints of your tasks, other algorithms like Qlearning, Whale, or SARSA might be considered for their respective strengths.

