Implementation Overview

Merge Sort Algorithm:

Merge sort is a divide-and-conquer sorting algorithm that recursively divides the input array into smaller subarrays, sorts them, and then merges them back together. The parallel version distributes these subarrays across multiple CPU cores for faster processing.

Algorithm Description:

- Divide-and-conquer sorting algorithm
- Divide-and-conquer sorting algorithms
- Merges sorted subarrays to produce final sorted array

Implementation Features:

- Sequential and parallel versions
- Performance monitoring and analysis
- Large dataset handling (1000-5000 elements)

Key Features of the Implementation:

- **Sequential Implementation**:
  - Pure Python implementation of merge sort
  - Memory usage tracking
  - Execution time measurement
  - Automatic dataset generation
  - Result verification
- **Parallel Implementation:**
  - Multiprocessing-based parallelization
  - Configurable number of threads
  - System resource monitoring
  - Performance metrics collection
  - Scalability analysis
- **Analysis Tools:**
  - Comprehensive performance metrics
  - Graphical visualization of results
  - CSV export of performance data
  - Speedup and efficiency calculations
  - System resource utilization tracking

Implementation Output Comparison

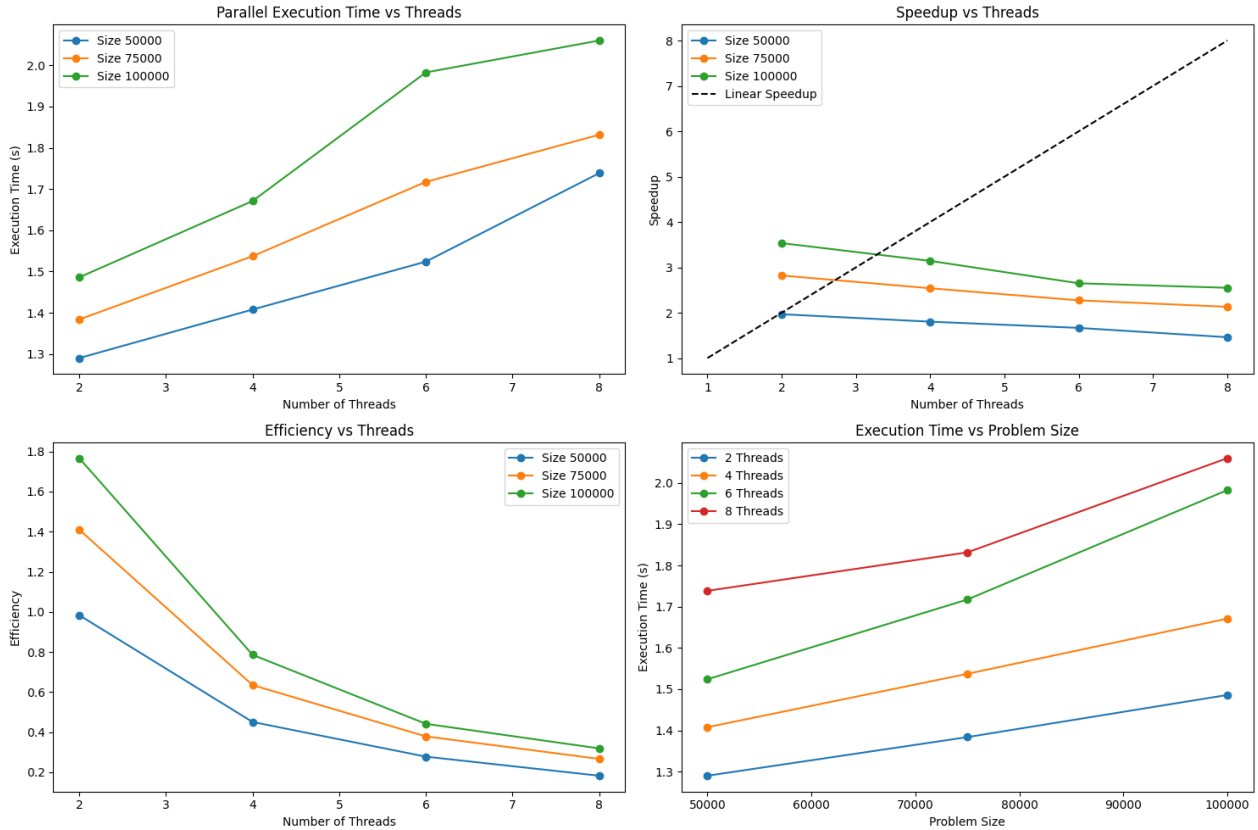
Execution Time Comparison:

Dataset Size	Execution time in Sequential Mode (seconds)	Avg. Execution time in Parallel Mode (seconds)
50000	1. 2475	0.3702
75000	1. 9535	0.4885
100000	2. 6361	0.5883

Performance Evaluation Metrics Collected:

- Execution Time
- Memory Usage
- CPU Utilization
- Speedup Ratio
- Parallel Efficiency
- Scaling Performance

Comparison Graphs:



File Structure:

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        24-02-2025     21:22                __pycache__
-a----        24-02-2025     21:21           3633 analyze_performance.py
-a----        24-02-2025     22:52         688865 input_data.txt
-a----        24-02-2025     21:19           5021 parallel_sort.py
-a----        24-02-2025     21:23         155498 performance_analysis.png
-a----        24-02-2025     21:23           1064 performance_results.csv
-a----        24-02-2025     21:18           3374 sequential_sort.py
```

Commands for running implementation programs:

- python .\sequential_sort.py
- python .\parallel_sort.py
- python .\analyze_performance.py

Implementation Output

Sequential Mode:

1. Dataset size: 50000
   Execution time: 1.2475 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    64     32.7 MiB     32.7 MiB           1       @profile
    65                                             def run_sorting(self):
    66                                                 """Execute sorting and measure performance"""
    67     32.7 MiB      0.0 MiB           1           if self.data is None:
    68     33.1 MiB      0.4 MiB           1               self.load_dataset()
    69
    70     33.1 MiB      0.0 MiB           1           print("\nStarting sequential merge sort...")
    71     33.1 MiB      0.0 MiB           1           start_time = time.time()
    72
    73                                                 # Convert to list for sorting
    74     35.0 MiB      1.9 MiB           1           data_list = self.data.tolist()
    75     37.8 MiB      2.8 MiB           1           sorted_data = self.merge_sort(data_list)
    76
    77     37.8 MiB      0.0 MiB           1           end_time = time.time()
    78     37.8 MiB      0.0 MiB           1           execution_time = end_time - start_time
    79
    80                                                 # Verify sorting
    81     37.8 MiB      0.0 MiB      100001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
    82
    83     37.8 MiB      0.0 MiB           1           print(f"\nExecution Results:")
    84     37.8 MiB      0.0 MiB           1           print(f"{'='*50}")
    85     37.8 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
    86     37.8 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
    87     37.8 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
    88     37.8 MiB      0.0 MiB           1           print(f"{'='*50}")
    89
    90     37.8 MiB      0.0 MiB           1           return execution_time, is_sorted
```

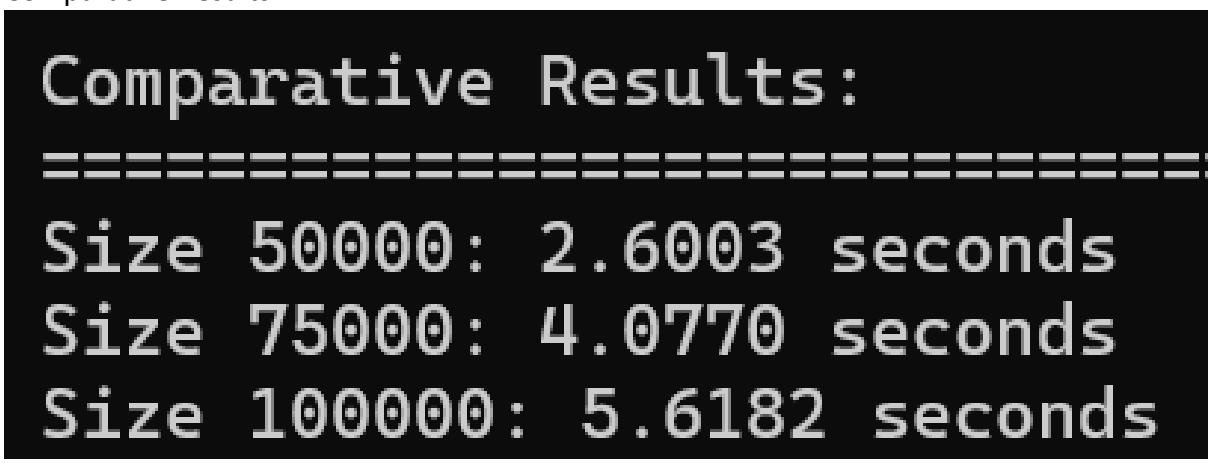
2. Dataset size: 75000
   Execution time: 1.9535 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    64     34.3 MiB     34.3 MiB           1       @profile
    65                                             def run_sorting(self):
    66                                                 """Execute sorting and measure performance"""
    67     34.3 MiB      0.0 MiB           1           if self.data is None:
    68     34.6 MiB      0.3 MiB           1               self.load_dataset()
    69
    70     34.6 MiB      0.0 MiB           1           print("\nStarting sequential merge sort...")
    71     34.6 MiB      0.0 MiB           1           start_time = time.time()
    72
    73                                                 # Convert to list for sorting
    74     36.9 MiB      2.4 MiB           1           data_list = self.data.tolist()
    75     37.6 MiB      0.7 MiB           1           sorted_data = self.merge_sort(data_list)
    76
    77     37.6 MiB      0.0 MiB           1           end_time = time.time()
    78     37.6 MiB      0.0 MiB           1           execution_time = end_time - start_time
    79
    80                                                 # Verify sorting
    81     37.6 MiB      0.0 MiB      150001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
    82
    83     37.6 MiB      0.0 MiB           1           print(f"\nExecution Results:")
    84     37.6 MiB      0.0 MiB           1           print(f"{'='*50}")
    85     37.6 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
    86     37.6 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
    87     37.6 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
    88     37.6 MiB      0.0 MiB           1           print(f"{'='*50}")
    89
    90     37.6 MiB      0.0 MiB           1           return execution_time, is_sorted
```

3. Dataset size: 100000
   Execution time: 2.6361 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    64     35.2 MiB     35.2 MiB           1       @profile
    65                                             def run_sorting(self):
    66                                                 """Execute sorting and measure performance"""
    67     35.2 MiB      0.0 MiB           1           if self.data is None:
    68     35.6 MiB      0.4 MiB           1               self.load_dataset()
    69
    70     35.6 MiB      0.0 MiB           1           print("\nStarting sequential merge sort...")
    71     35.6 MiB      0.0 MiB           1           start_time = time.time()
    72
    73                                                 # Convert to list for sorting
    74     38.7 MiB      3.0 MiB           1           data_list = self.data.tolist()
    75     38.7 MiB      0.1 MiB           1           sorted_data = self.merge_sort(data_list)
    76
    77     38.7 MiB      0.0 MiB           1           end_time = time.time()
    78     38.7 MiB      0.0 MiB           1           execution_time = end_time - start_time
    79
    80                                                 # Verify sorting
    81     38.7 MiB      0.0 MiB      200001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
    82
    83     38.7 MiB      0.0 MiB           1           print(f"\nExecution Results:")
    84     38.7 MiB      0.0 MiB           1           print(f"{'='*50}")
    85     38.7 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
    86     38.7 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
    87     38.7 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
    88     38.7 MiB      0.0 MiB           1           print(f"{'='*50}")
    89
    90     38.7 MiB      0.0 MiB           1           return execution_time, is_sorted
```

Comparative Results:



Parallel Mode:

1. Dataset size: 50000
    a. Threads: 2
    Memory used: 36.68 MB
    Execution time: 0.3032 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     33.3 MiB     33.3 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     33.3 MiB      0.0 MiB           1           if self.data is None:
    95     33.8 MiB      0.5 MiB           1               self.load_dataset()
    96
    97     33.8 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     33.8 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     36.7 MiB      2.8 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     36.7 MiB      0.0 MiB           1           end_time = time.time()
   103     36.7 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     36.7 MiB      0.0 MiB      100001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     36.7 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     36.7 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     36.7 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     36.7 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     36.7 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     36.7 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     36.7 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     36.7 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     36.7 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     36.7 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     36.7 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     36.7 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     36.7 MiB      0.0 MiB           1           return execution_time, is_sorted
```

    b. Threads: 4
    Memory used: 37.90 MB
    Execution time: 0.3583 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     35.1 MiB     35.1 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     35.1 MiB      0.0 MiB           1           if self.data is None:
    95     35.3 MiB      0.2 MiB           1               self.load_dataset()
    96
    97     35.3 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     35.3 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     37.9 MiB      2.6 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     37.9 MiB      0.0 MiB           1           end_time = time.time()
   103     37.9 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     37.9 MiB      0.0 MiB      100001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     37.9 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     37.9 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     37.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     37.9 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     37.9 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     37.9 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     37.9 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     37.9 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     37.9 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     37.9 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     37.9 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     37.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     37.9 MiB      0.0 MiB           1           return execution_time, is_sorted
```

c. Threads: 6
Memory used: 38.03 MB
Execution time: 0.4493 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     35.6 MiB     35.6 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     35.6 MiB      0.0 MiB           1           if self.data is None:
    95     36.0 MiB      0.3 MiB           1               self.load_dataset()
    96
    97     36.0 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     36.0 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     38.0 MiB      2.1 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     38.0 MiB      0.0 MiB           1           end_time = time.time()
   103     38.0 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     38.0 MiB      0.0 MiB      100001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     38.0 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     38.0 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     38.0 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     38.0 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     38.0 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     38.0 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     38.0 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     38.0 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     38.0 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     38.0 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     38.0 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     38.0 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     38.0 MiB      0.0 MiB           1           return execution_time, is_sorted
```

d. Threads: 8
Memory used: 38.19 MB
Execution time: 0.4706 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     35.9 MiB     35.9 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     35.9 MiB      0.0 MiB           1           if self.data is None:
    95     36.0 MiB      0.1 MiB           1               self.load_dataset()
    96
    97     36.0 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     36.0 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     38.2 MiB      2.2 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     38.2 MiB      0.0 MiB           1           end_time = time.time()
   103     38.2 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     38.2 MiB      0.0 MiB      100001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     38.2 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     38.2 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     38.2 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     38.2 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     38.2 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     38.2 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     38.2 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     38.2 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     38.2 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     38.2 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     38.2 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     38.2 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     38.2 MiB      0.0 MiB           1           return execution_time, is_sorted
```

2. Dataset size: 75000
    a. Threads: 2
       Memory used: 39.32 MB
       Execution time: 0.3514 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     35.9 MiB     35.9 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     35.9 MiB      0.0 MiB           1           if self.data is None:
    95     36.2 MiB      0.3 MiB           1               self.load_dataset()
    96
    97     36.2 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     36.2 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     39.3 MiB      3.1 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     39.3 MiB      0.0 MiB           1           end_time = time.time()
   103     39.3 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     39.3 MiB      0.0 MiB      150001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     39.3 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     39.3 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     39.3 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     39.3 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     39.3 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     39.3 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     39.3 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     39.3 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     39.3 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     39.3 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     39.3 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     39.3 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     39.3 MiB      0.0 MiB           1           return execution_time, is_sorted
```

    b. Threads: 4
       Memory used: 40.03 MB
       Execution time: 0.4593 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     36.3 MiB     36.3 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     36.3 MiB      0.0 MiB           1           if self.data is None:
    95     36.5 MiB      0.1 MiB           1               self.load_dataset()
    96
    97     36.5 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     36.5 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     40.0 MiB      3.5 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     40.0 MiB      0.0 MiB           1           end_time = time.time()
   103     40.0 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     40.0 MiB      0.0 MiB      150001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     40.0 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     40.0 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     40.0 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     40.0 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     40.0 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     40.0 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     40.0 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     40.0 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     40.0 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     40.0 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     40.0 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     40.0 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     40.0 MiB      0.0 MiB           1           return execution_time, is_sorted
```

c. Threads: 6
Memory used: 39.50 MB
Execution time: 0.5314 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     36.5 MiB     36.5 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     36.5 MiB      0.0 MiB           1           if self.data is None:
    95     36.9 MiB      0.4 MiB           1               self.load_dataset()
    96
    97     36.9 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     36.9 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     39.5 MiB      2.6 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     39.5 MiB      0.0 MiB           1           end_time = time.time()
   103     39.5 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     39.5 MiB      0.0 MiB      150001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     39.5 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     39.5 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     39.5 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     39.5 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     39.5 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     39.5 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     39.5 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     39.5 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     39.5 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     39.5 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     39.5 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     39.5 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     39.5 MiB      0.0 MiB           1           return execution_time, is_sorted
```

d. Threads: 8
Memory used: 39.82 MB
Execution time: 0.6119 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     37.1 MiB     37.1 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     37.1 MiB      0.0 MiB           1           if self.data is None:
    95     37.2 MiB      0.1 MiB           1               self.load_dataset()
    96
    97     37.2 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     37.2 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     39.8 MiB      2.6 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     39.8 MiB      0.0 MiB           1           end_time = time.time()
   103     39.8 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     39.8 MiB      0.0 MiB      150001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     39.8 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     39.8 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     39.8 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     39.8 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     39.8 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     39.8 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     39.8 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     39.8 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     39.8 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     39.8 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     39.8 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     39.8 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     39.8 MiB      0.0 MiB           1           return execution_time, is_sorted
```

3. Dataset size: 100000
    a. Threads: 2
       Memory used: 40.89 MB
       Execution time: 0.4049 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     36.9 MiB     36.9 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     36.9 MiB      0.0 MiB           1           if self.data is None:
    95     37.3 MiB      0.4 MiB           1               self.load_dataset()
    96
    97     37.3 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     37.3 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     40.9 MiB      3.5 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     40.9 MiB      0.0 MiB           1           end_time = time.time()
   103     40.9 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     40.9 MiB      0.0 MiB      200001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     40.9 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     40.9 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     40.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     40.9 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     40.9 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     40.9 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     40.9 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     40.9 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     40.9 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     40.9 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     40.9 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     40.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     40.9 MiB      0.0 MiB           1           return execution_time, is_sorted
```

    b. Threads: 4
       Memory used: 41.29 MB
       Execution time: 0.5206 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     36.9 MiB     36.9 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     36.9 MiB      0.0 MiB           1           if self.data is None:
    95     37.2 MiB      0.2 MiB           1               self.load_dataset()
    96
    97     37.2 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     37.2 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     41.3 MiB      4.1 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     41.3 MiB      0.0 MiB           1           end_time = time.time()
   103     41.3 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     41.3 MiB      0.0 MiB      200001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     41.3 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     41.3 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     41.3 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     41.3 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     41.3 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     41.3 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     41.3 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     41.3 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     41.3 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     41.3 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     41.3 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     41.3 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     41.3 MiB      0.0 MiB           1           return execution_time, is_sorted
```

c. Threads: 6
Memory used: 40.93 MB
Execution time: 0.6902 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     36.9 MiB     36.9 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     36.9 MiB      0.0 MiB           1           if self.data is None:
    95     37.3 MiB      0.4 MiB           1               self.load_dataset()
    96
    97     37.3 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     37.3 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     40.9 MiB      3.6 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     40.9 MiB      0.0 MiB           1           end_time = time.time()
   103     40.9 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     40.9 MiB      0.0 MiB      200001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     40.9 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     40.9 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     40.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     40.9 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     40.9 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     40.9 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     40.9 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     40.9 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     40.9 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     40.9 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     40.9 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     40.9 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     40.9 MiB      0.0 MiB           1           return execution_time, is_sorted
```

d. Threads: 8
Memory used: 41.20 MB
Execution time: 0.7376 seconds

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91     37.2 MiB     37.2 MiB           1       @profile
    92                                             def run_sorting(self):
    93                                                 """Execute parallel sorting and measure performance"""
    94     37.2 MiB      0.0 MiB           1           if self.data is None:
    95     37.7 MiB      0.4 MiB           1               self.load_dataset()
    96
    97     37.7 MiB      0.0 MiB           1           print(f"\nStarting parallel merge sort with {self.num_threads} threads...")
    98     37.7 MiB      0.0 MiB           1           start_time = time.time()
    99
   100     41.2 MiB      3.5 MiB           1           sorted_data = self.parallel_merge_sort(self.data.tolist())
   101
   102     41.2 MiB      0.0 MiB           1           end_time = time.time()
   103     41.2 MiB      0.0 MiB           1           execution_time = end_time - start_time
   104
   105     41.2 MiB      0.0 MiB      200001           is_sorted = all(sorted_data[i] <= sorted_data[i+1] for i in range(len(sorted_data)-1))
   106     41.2 MiB      0.0 MiB           1           sys_info = self.get_system_info()
   107
   108     41.2 MiB      0.0 MiB           1           print(f"\nExecution Results:")
   109     41.2 MiB      0.0 MiB           1           print(f"{'='*50}")
   110     41.2 MiB      0.0 MiB           1           print(f"Dataset size: {self.size}")
   111     41.2 MiB      0.0 MiB           1           print(f"Number of threads: {self.num_threads}")
   112     41.2 MiB      0.0 MiB           1           print(f"Physical cores: {sys_info['physical_cores']}")
   113     41.2 MiB      0.0 MiB           1           print(f"Total cores: {sys_info['total_cores']}")
   114     41.2 MiB      0.0 MiB           1           print(f"CPU frequency: {sys_info['cpu_freq']:.2f} MHz")
   115     41.2 MiB      0.0 MiB           1           print(f"Memory used: {sys_info['memory_used']:.2f} MB")
   116     41.2 MiB      0.0 MiB           1           print(f"Execution time: {execution_time:.4f} seconds")
   117     41.2 MiB      0.0 MiB           1           print(f"Sorting verified: {is_sorted}")
   118     41.2 MiB      0.0 MiB           1           print(f"{'='*50}")
   119
   120     41.2 MiB      0.0 MiB           1           return execution_time, is_sorted
```

Comparative Results: