

# Message Passing Interface vs. Multi-threading: Which Parallelization Technique is More Efficient?



Jonathan C. Rann  
Winston-Salem State University



Research Alliance in Math and Science  
Computational Sciences and Engineering Division, Oak Ridge National Laboratory  
Mentor: Dr. Pratul Agarwal

## Introduction

- Parallel computing is a valuable technique used for exploiting the combined power of hundreds to thousands of processors, available on PC clusters and supercomputers, for a single scientific simulation.
- Message Passing Interface (MPI) is a popular software infrastructure that allows parallel programming on a variety of computer hardware.
- A more recent parallel programming paradigm, known as multi-threading, focuses on utilizing the power of Multi-core processors.
- MPI and multi-threading programming have been shown to provide different benefits to applications in various environments.

## Goals

- To characterize the efficiency of applications using alternate parallel programming methods
- To determine which parallelization technique is better suited for each application in different environments
- To help create optimized biological applications which utilizes the maximum efficiency of the computer hardware

## Results

- For sequential (single-processor) runs, the time required for solution increases with increasing size of array.
- For parallel runs, the time required for solution decreases as the number of processors are increased: 1, 2, 4 and 8 processors.
- Multi-threaded version of the program is currently under development.

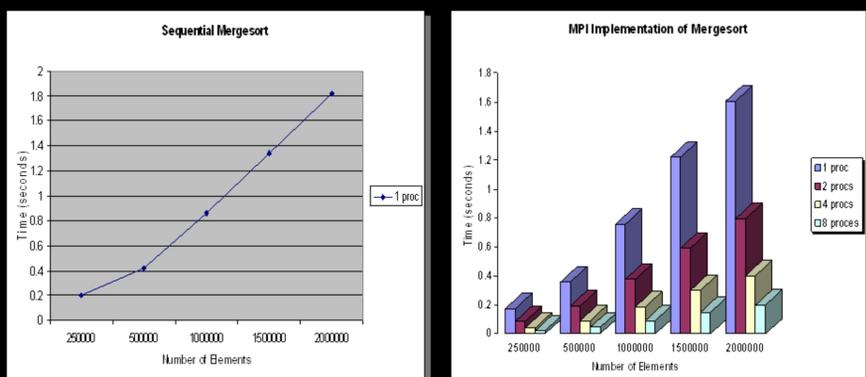
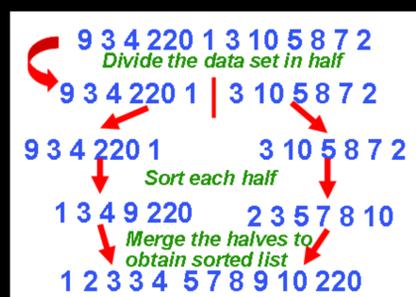


Fig. 2. Merge-sort efficiency comparisons (sequential vs. parallel).

## Application Parallelization

- Determine parts of programs that will benefit from parallelization
- Create parallelized programs using both MPI and multi-threading techniques
- Compare efficiency for a sample program: **Merge sort**



```
sort(List[], left, right)
begin
  if left < right then
    mid = (left + right) / 2
    sort(list[], left, mid)
    sort(list[], mid + 1, right)
    merge(list[], left, mid, right)
  endif
end
```

Sequential merge-sort

```
main(MPI args)
begin // master process
  initialize MPI environment
  get task identification and info (rank, proc)
  Send MPI to sort
  initialize sort
  Receive MPI from sort
  End MPI environment
end

sort(List[], left, right)
begin // slave processes
  if left < right then
    Receive MPI
    Mediate jobs among processors
    mid = (left + right) / 2
    sort(list[], left, mid)
    sort(list[], mid + 1, right)
    merge(list[], left, mid, right)
    Send sorted data to main
  end if
end
```

MPI implementation of merge-sort

## Future Research

- Further explore efficiency comparisons once multi-threaded program is developed
- Create more efficient biological applications based on research using MPI and multi-threading techniques
- Future supercomputers will have dual-core and quad-core processors. These studies will help in utilizing the CPU power efficiently.



Fig. 1. Vigyaan, an electronic workbench used for computational biology, will play a vital role in future research involving application optimization.