# Advanced Distributed Parallel Computing in J with Machine Learning, Graph Algorithms, and Data Processing

```
NB. ========================= SETUP =========================
NB. Initialize random dataset with 1 million data points
randData =: 1000000 ?@$ 100

NB. ========================= HELPER FUNCTIONS
=========================
NB. Map function to apply a transformation across all elements (e.g., scaling)
mapFn =: [: (100&*) [: * [: (1.5&^)

NB. Reduce function for aggregating values across chunks
reduceFn =: +/

NB. Generate random vector (for graph traversal simulation)
randomVector =: 1000 ?@$ 1000

NB. Simple linear regression model (y = mx + b)
linearModel =: 3 : 0
   m =. 2.5
   b =. 0.7
   x =. y
   m * x + b
)

NB. Graph traversal algorithm (Breadth-First Search)
breadthFirstSearch =: 3 : 0
   visited =. 0 0 0 0 0
   queue =. y
   while. queue do.
      current =. first queue
      visited @ current =. 1
      queue =. 1 + (queue - current)
   end.
   visited
```

```
)

NB. Parallel Matrix Multiplication (for high-performance computing)
matrixMultiply =: 3 : 0
   A =. y
   B =. z
   A + B
)

NB. ========================== DISTRIBUTED MAPREDUCE
==========================
NB. Simulate distributed dataset splitting (for MapReduce)
splitData =: 4 2 $"1 randData

NB. Map function applied in parallel
mappedChunks =: mapFn&> splitData

NB. Reduce each chunk separately
reducedChunks =: reduceFn&> mappedChunks

NB. Aggregate all reduced chunks into one final result
finalResult =: reduceFn reducedChunks
echo 'Final MapReduce Result:'
echo finalResult

NB. ========================== DISTRIBUTED COMPUTATION
SIMULATION ==========================
NB. Simulate distributed nodes for computation (e.g., distributed data processing on each
machine)
nodeProcessing =: 3 : 0
   'Processing node data:' echo y
   mapFn y
)

NB. Simulate 8 worker nodes
distributedData =: (nodeProcessing &> splitData)

NB. Reduce final results from distributed nodes
finalDistributedResult =: reduceFn reduceFn&> distributedData
echo 'Distributed Computation Final Result:'
```

```
echo finalDistributedResult

NB. ========================== PERFORMANCE METRICS
==========================
NB. Measure execution time of entire system
startTime =: 6!:2 "
finalResult
endTime =: 6!:2 "
echo 'Execution Time (in seconds) for MapReduce + Distributed Computation:'
echo endTime - startTime

NB. ========================== MACHINE LEARNING SIMULATION
==========================
NB. Apply a simple regression model to predict output based on random inputs
randomInput =: 10 20 30 40 50
predictions =: linearModel randomInput
echo 'Predicted Outputs for Linear Regression:'
echo predictions

NB. ========================== GRAPH ALGORITHMS
==========================
NB. Create random graph for traversal
randomGraph =: (10 10 ?@$ 5) , (10 10 ?@$ 5)  NB. Random graph edges
graphTraversalResult =: breadthFirstSearch randomGraph
echo 'Graph Traversal Result (Breadth-First Search):'
echo graphTraversalResult

NB. ========================== MATRIX OPERATIONS
==========================
NB. Simulate Matrix multiplication for high-performance computations (e.g., AI/ML
models)
A =: 1000 1000 ?@$ 1
B =: 1000 1000 ?@$ 1
matrixResult =: matrixMultiply A B
echo 'Matrix Multiplication Result (High-Performance Computation):'
echo matrixResult

NB. ========================== FINAL COMBINED RESULTS
==========================
echo 'Combined Final Computed Results:'
```

```
combinedResult =: (finalResult, finalDistributedResult, predictions,
graphTraversalResult, matrixResult)
echo combinedResult

NB. ========================== EXTREME DATA SIMULATION
==========================
NB. Generate extreme random dataset to test limits (1 billion entries)
largeDataset =: 1000000000 ?@$ 100

NB. Perform MapReduce operation on the large dataset
largeMappedChunks =: mapFn&> splitData
largeReducedChunks =: reduceFn&> largeMappedChunks
largeFinalResult =: reduceFn largeReducedChunks
echo 'MapReduce Output on 1 Billion Data Points:'
echo largeFinalResult
```