RESEARCH ARTICLE

Cost-Optimized Data Partitioning for Parallel Quicksort on Heterogeneous Clusters using Mixed Integer Programming

Joeniño Cainday and Dr. Junar Landicho

Department of Computer Science, University of Science and Technology of Southern Philippines, Cagayan De Oro City, Philippines

ARTICLE HISTORY
Compiled May 4, 2025

ABSTRACT
This research addresses the optimization of data partitioning for parallel computations in heterogeneous distributed systems, with the dual objective of minimizing financial expenditure and execution time. We develop a Mixed Integer Programming (MIP) model that incorporates critical system constraints including CPU speed, memory capacity, memory latency, network latency, and budget limitations. Our model replaces the heuristic partitioning phase of a parallel quicksort algorithm, which serves as our performance benchmark. Utilizing synthetic datasets representing both workload and node specifications, our simulations focus on pre-execution data partitioning decisions. The goal is to provide a framework for financial and time optimization within cloud computing environments, where efficient resource allocation and budget management are essential. Results demonstrate the effectiveness of our MIP-based approach in achieving improved load distribution and reduced runtime compared to traditional heuristics.

KEYWORDS
Data Partitioning; Mixed Integer Programming; Heterogeneous Distributed Systems; Budget Constraints; Makespan Minimization

1. Introduction

The widespread adoption of cloud computing and serverless architectures has increased the importance of efficient resource allocation in distributed systems. This research addresses the challenge of statically partitioning data across heterogeneous computing environments, where nodes vary in processing speed, memory capacity, network latency, and monetary implications.

We formulate data partitioning as an NP-hard combinatorial optimization problem aimed at minimizing execution time (makespan) while adhering to defined constraints. Our approach replaces traditional heuristic-based partitioning in parallel sorting algorithms with a novel Mixed Integer Programming (MIP) model designed to capture comprehensive system constraints for more balanced, resource-aware data distributions.

1.1. Background and Motivation

Distributed computing enables parallel processing of large-scale datasets across multiple nodes. However, in heterogeneous systems with diverse computational capabilities,

simplistic data partitioning strategies often result in load imbalances and inefficient resource utilization [1]. Common approaches like uniform distribution or partitioning based solely on heuristics typically fail to consider system compositions in cloud environments.

Cloud computing introduces new complexities to data partitioning. These environments offer flexibility with pay-as-you-go pricing models where computing resources are available on demand [5]. This necessitates re-evaluating traditional partitioning methodologies to explicitly consider economic factors alongside performance metrics. Effective strategies must balance high performance with budget limitations—minimizing both computational time and overall expenditure for cloud resource utilization.

While Monga and Lodhi [53] demonstrated benefits of heterogeneity-aware allocation, their model primarily considered CPU speed alone. Our work introduces a more sophisticated MIP-based model that integrates multiple system attributes to achieve globally optimized data distributions, providing a more realistic and budget-conscious approach to optimized data processing.

1.2. Cost Considerations in Heterogeneous Parallel Processing

This paper specifically addresses statically partitioning large datasets before executing parallel Quicksort across simulated heterogeneous environments. Our primary objective is to minimize the overall execution time (makespan) while ensuring we satisfy the defined constraints. As a secondary objective, we aim to identify the most budget-effective solution with an equivalent makespan, using a lexicographic optimization approach.

In this context, cost refers to the pricing of cloud compute nodes. While higher-priced instances often imply better performance, this is not guaranteed, as pricing and performance can vary across vendors. Each simulated node has synthetic attributes including processing speed, memory capacity, network characteristics, and pricing models inspired by real cloud offerings, reflecting typical variations in heterogeneous cloud deployments.

Allocating data across heterogeneous resources inherently involves performance-cost tradeoffs [48]. As highlighted in previous research, balancing processing time with financial implications when assigning datasets to different machines is critical. Effective data partitioning can significantly reduce both runtime and monetary cost, while suboptimal partitioning leads to inflated expenses or resource limitations like memory overflows. Our controlled synthetic environment enables repeatable experiments independent of real-world cloud variability.

MIP provides a powerful mathematical framework for complex optimization problems involving both discrete and continuous decision variables subject to linear constraints [8]. This capability makes MIP well-suited for modeling cost-aware data partitioning in heterogeneous systems:

- Discrete variables represent partitioning decisions (assigning specific data blocks to particular nodes)
- Continuous variables model resource utilization levels and associated costs
- The objective function represents total financial expenditure
- Constraints reflect resource limitations and system characteristics

Through this approach, MIP effectively finds optimal or near-optimal partitioning strategies that minimize expenditure while satisfying performance requirements.

1.3. Research Contributions

This paper contributes the following:

(1) A MIP formulation for initial data partitioning in parallel quicksort on heterogeneous systems;
(2) An integration of the MIP model into an existing heterogeneous sorting framework;
(3) A performance evaluation using synthetic datasets with extended metrics;
(4) A reproducible experimental setup and discussion of directions for future work.

2. Related Work

Heterogeneous scheduling and data allocation have been studied in various contexts. [48] considered dataset allocation across geo-distributed clouds with two objectives (processing time and cost) [48]. They formulated a linear program to place data blocks on VMs to minimize weighted sum of time and cost, demonstrating Pareto trade-offs. Similarly, [49] used an integer linear program for data assignment in a hybrid heterogeneous processing environment. These works focus on resource assignment rather than within-job data partitioning, but they highlight the utility of mathematical programming for cloud costs.

Classic scheduling theory demonstrates that even simple load-balancing on unrelated machines is NP-hard [50]. For instance, scheduling jobs to minimize makespan in admits a 2-approximation algorithm but cannot be solved optimally in polynomial time beyond trivial cases [50]. This complexity motivates the treatment of cost-aware data partitioning as a combinatorial optimization problem, which is closely related to but with additional cost considerations. This can be extended to multi-objective formulations (e.g., cost and makespan), though even the single-objective version is NP-hard, reducible to scheduling on unrelated parallel machines [50].

In the domain of parallel sorting, many algorithms assume a homogeneous machine model. For instance, Parallel Sorting by Regular Sampling (PSRS) chooses pivots to create equally-sized partitions [51], and typical benchmarks use randomly-generated data with uniform or other distributions [51]. These benchmarks (e.g. uniform 32-bit integer inputs) guide our synthetic data choices. However, PSRS and related methods do not account for cost or heterogeneous speeds.

In big-data systems like Spark, dynamic partitioning and scheduling algorithms have been proposed. [52], for example, developed a dynamic partitioning strategy for intermediate Spark data to mitigate skew, and a greedy scheduling method that considers node speed [52]. They find that balanced partitioning significantly lowers completion time [52]. Our work differs by focusing on static initial partitioning with explicit cost metrics, rather than in-job rebalancing. To the best of our knowledge, prior work has not explicitly applied MIP to cost-aware static data partitioning in heterogeneous cloud sort workloads.

We build upon the model proposed by Monga and Lodhi [53], which partitions datasets across heterogeneous nodes proportionally to CPU speed to achieve near-optimal load balance in parallel sorting. However, their approach does not account for cost or network latency, both of which are critical in modern cloud and serverless computing. We generalize this problem and propose a Mixed Integer Programming (MIP) formulation that jointly optimizes cost and performance under multiple realistic constraints. We also compare against their CPU-speed-based heuristic and other baseline

methods to evaluate trade-offs.

3. Proposed Methodology

This section presents our approach combining Mixed Integer Programming (MIP) for optimal data partitioning with the parallel Quicksort algorithm proposed by Monga and Lodhi [53].

3.1. Problem Formulation

We now formalize the partitioning problem based on the system characteristics introduced in Section 1.3. To ensure full control over system parameters and facilitate reproducibility, we operate within a synthetic environment. Each node is defined by parameters such as processing speed, memory capacity, network latency, bandwidth, and cost, inspired by real-world cloud configurations (e.g., AWS, GCP). This abstraction enables rigorous evaluation of partitioning strategies without relying on live infrastructure.

3.1.1. System Model

We partition the dataset into contiguous blocks of size d, assigning each block to a unique node. Consider a heterogeneous cluster with N nodes, where each node i is characterized by the following parameters:

- d_i: Assigned data volume (in units of d)
- r_i: Processing rate (in d/t — data units per time unit)
- m_i: Available memory capacity (in units of m)
- ℓ_i: Network latency (in time units t)
- b_i: Network bandwidth (in d/t — data units per time unit)
- u_i: Usage billing rate (in c/t — cost units per time unit)

The units provided above are illustrative and can be adjusted depending on the application context. For example, processing speed may be expressed in records per millisecond, memory in MB, or cost in USD/hour. In this work, we use normalized or synthetic units to model relative performance and cost differences between nodes, without binding the system to a specific infrastructure or currency.

3.1.2. Derived Parameters

We derive additional parameters from the node characteristics to facilitate the MIP formulation:

3.1.2.1. Processing Time.

$$x_i = \frac{d_i}{r_i} \tag{1}$$

where x_i is the time required by node i to process its assigned data volume d_i.

3.1.2.2. Transfer Time.

$$y_i = \ell_i + \frac{d_i}{b_i} \tag{2}$$

where y_i is the communication overhead for node i, computed from latency ℓ_i and bandwidth b_i.

3.1.2.3. Total Cost.

$$w = \sum_{i=1}^{N} u_i \cdot x_i \tag{3}$$

where w denotes the cumulative cost of utilizing all selected nodes.

3.1.3. Optimization Objectives

The primary objective of our MIP model is to minimize the total execution time (makespan) of the parallel sorting process. This ensures that the overall completion time is minimized, taking into account both computation and communication overheads. Formally, the primary objective is defined as:

$$\min z = \max_{i=1}^{N}(x_i + y_i) \tag{4}$$

where z represents the maximum time taken by any node i to complete its assigned data processing and communication tasks. This formulation captures the makespan of the entire parallel sorting operation, ensuring that the slowest node determines the overall completion time.

In addition to minimizing makespan, we aim to choose the most cost-effective solution among equivalent combinations. To achieve this, we introduce a secondary objective that minimizes the total financial expenditure. This is incorporated into the model using a lexicographic optimization approach:

$$\min \ z + \varepsilon \cdot w \tag{5}$$

In this formulation, w represents the cumulative cost of utilizing the selected nodes, and ε is set to a negligible value (e.g., 10^{-6}) to prioritize makespan minimization while breaking ties in favor of cost efficiency. This approach ensures that among solutions with equivalent makespan, the one with the lowest cost is selected.

3.1.4. Constraint Definitions

The proposed MIP model is subject to several constraints that reflect system limitations and ensure a feasible allocation of data across nodes. These constraints incorporate resource boundaries (e.g., memory and budget), performance considerations (e.g., makespan), and completeness of the partitioning scheme.

3.1.4.1. Makespan Constraint.

$$z \geq \frac{d_i}{r_i} + \ell_i + \frac{d_i}{b_i} \quad \forall i \in \{1, \ldots, N\} \tag{6}$$

This constraint ensures that the total execution time, represented by the variable z, is at least as large as the time taken by any individual node i to both process its assigned data and transfer it to the next stage. It combines computation time $\frac{d_i}{r_i}$ with communication latency ℓ_i and data transfer time $\frac{d_i}{b_i}$.

3.1.4.2. Memory Constraint.

$$d_i \leq m_i \quad \forall i \in \{1, \ldots, N\} \tag{7}$$

Each node has a limited memory capacity m_i, and this constraint ensures that the volume of data d_i assigned to node i does not exceed its available memory.

3.1.4.3. Coverage Constraint.

$$\sum_{i=1}^{N} d_i = D \tag{8}$$

This constraint enforces full data allocation: the entire dataset of size D must be partitioned and distributed among the N available nodes without omission or duplication.

3.1.4.4. Non-negativity Constraint.

$$d_i \geq 0 \quad \forall i \in \{1, \ldots, N\} \tag{9}$$

This standard constraint ensures that each node receives a non-negative volume of data, reflecting the physical impossibility of assigning negative quantities.

3.1.4.5. Budget Constraint (Optional).

$$0 \leq \sum_{i=1}^{N} u_i \cdot d_i \leq B \tag{10}$$

In addition to node-specific parameters, we optionally define a user-defined input B, representing the maximum allowable expenditure. This parameter must be a non-negative number, with a default value of 0. Including this constraint models scenarios where financial limitations must be respected.