A Brief of Distributed Data Processing

Clemente Izurieta
Montana State University
Gianforte School of Computing
Bozeman, MT, USA
clemente.izurieta@montana.edu

Nate Woods
Montana State University
Gianforte School of Computing
Bozeman, MT, USA
me@bign8.info

Ann Marie Reinhold
Montana State University
Gianforte School of Computing
Bozeman, MT, USA
reinhold@montana.edu

Abstract—**Distributed data processing is a cornerstone in modern cloud and edge computing environments because of its ability to handle large amounts of information that can overwhelm a single computer. However, the ontogeny of research in the field of distributed data processing remains poorly characterized. Therefore, we reviewed 70 publications discussing distributed data processing. Distributed processing systems is an active area of research with publications increasing in numbers since the early 2000s. The most salient topics in distributed processing systems were affiliated with system architecture and programming paradigms. However, researchers lack standard metrics for reporting throughput, hampering the comparison of existing studies. This study is a first step towards characterizing this field of research and identifying important areas of opportunity.**

Keywords—mapping study, data processing, distributed processing systems.

I. Introduction

The volume and velocity of big data pose significant challenges to the software engineering community. The rates at which data are being collected, stored, and processed in data-center and cloud-computing environments have grossly outpaced the rate at which hardware can be developed [5]. The current wave of data production has software designers playing catch-up with the volume of data needing to be processed.

Because distributed systems are capable of handling large amounts of data, software engineers have proposed several techniques to advance distributed data processing. However, these techniques have been developed without a thorough review of needs and trends in the field of software engineering. Here, we take a first step towards this review by addressing the following three goals:

- Goal 1 (G1): Assess and review areas of research in distributed data processing.
- Goal 2 (G2): Aggregate and report the throughput of distributed processing systems researched.
- Goal 3 (G3): Investigate trends in the primary literature to explore how the published research has documented research attention and changes.

A. Terminology

Throughout this manuscript, we reference foundational terms used in the distributed processing domain.

1) Data: A *datum* is a single piece of information or a fact that can also be referred to as a record or a sample. The pluralization of a datum is data, and a set of facts is also known as a set of data or a data set.

2) Worker: A *worker* is a single running process that is capable of handling a given task. Workers can also be called processors or nodes depending on the context, but for the purposes of this discussion, these terms can be considered equivalent. Here, we use the term "node" for consistency.

3) Batch Processing: Batch processing breaks data processing into separate gathering and processing phases. This enables software designers to isolate expensive processing from the critical path of data acquisition.

4) Stream Processing: Stream processing handles samples in real time, providing more frequent and incremental results than batch processing. This is achieved by allowing workers to maintain state from previous inputs and treating additional inputs as incremental changes to the current state.

B. Prior Work

No existing studies of distributed data processing cover both stream and batch processing in detail. Stream processing has been researched more extensively than batch processing. For instance, a systematic literature review of distributed data processing empirically characterized stream programs and contributed valuable details of stream processing programs to guide the design of stream processing frameworks [7]. However, batch processing techniques were not characterized in this work. No other studies provided ample treatment of batch processing, thereby leaving a gap in the literature on distributed data processing. Other mapping studies focused on distributed data processing within particular subdisciplines of computing (e.g., Big Data [1] and Internet of Things [2]). For instance, Akoka et al. [1] discuss Big Data, where distributed data processing was used to help process data. Similarly, Alkhabbas et al. [2] discuss the characterization of the Internet of Things (IoT) where distributed data processing focuses on *things* that have physical representation in the real world. However, we were unable to find a systematic investigation of distributed data processing across computing subdisciplines.

Critical gaps in the existing body of research remain. Our work fills these gaps by (1) covering a breadth that is not found in prior work including both batch and stream processing techniques; (2) ensuring our study is agnostic across computing subdisciplines (e.g., Big Data); (3) using *Natural Language Processing* (NLP) to impart objectivity; and (4) normalizing

performance metrics across studies to enable comparisons across works that no other study provides.

II. Approach

We reviewed the primary literature on the field of distributed data processing and the ontogeny of this field from 2002-2019 following the guidelines provided by Budgen et al. [4].

A. Literature Search Criteria and Strategy

We searched the IEEE Xplore®[1], Association for Computing Machinery Digital Library[2] and Google Scholar[3] for publications on the topic of distributed data processing. These sites provided indexed results on topical publications and ensured our search corpus covered a breadth of publications in distributed data processing.

To focus our review on the most salient and topical areas, we defined our search string as follows: ("distributed data" OR "data processing") AND ("processing pattern" OR "algorithm design"). These terms characterize the key concepts of distributed data processing, design patterns, and algorithmic design better than the blanket term "stream processing."

B. Selection Criteria

The pruning of the corpus was done using stringent inclusion and exclusion criteria. We included papers that discuss distributed data processing, were published between 2002-2019, were written in English, and contained experimental research. We excluded publications that used distributed data processing but did not research it directly, and eliminated publications that were not peer-reviewed. Using these criteria, we selected a total of 70 publications.

C. Identifying Contribution Areas and Common Terms

We performed a comprehensive review of all 70 papers by first reading them from cover-to-cover. We then determined the contribution areas in each publication to determine the scope of topics in the corpus using a manual coding approach. We coded the contribution areas of the publications by manually documenting the themes and topics described in the *Abstract, Introduction, Results, and Conclusion* sections of each paper. Using these documents, we then grouped related publications based on the topics each publication covered (Table I).

We next analyzed the abstracts from every paper using NLP. We strategically selected abstracts for NLP analysis because abstracts contain the most salient topics and important keywords that capture and characterize the most important topics in a study. Using WEKA-3.8.4 [6], we sanitized all words in each abstract by removing stop words using the English Dictionary from Natural Language Tool Kit [3]. We manually stemmed remaining words because automated stemming algorithms performed poorly on the technically written abstracts. We defined the set of common terms as the list of unique words present in 20% of abstracts. We calculated the frequency of each of these terms in each abstract in the corpus.

TABLE I
Publication Topics Identified via Manual Coding. Article Id corresponds to the unique article identifier in the corpus

Topic	Articl Id
System Design	
Network	25, 44
Dynamic	6, 8, 24, 41
Architecture	9, 29, 48
Performance	45, 67
Edge Computing	14, 30, 57
Single Processor	33
Extend Architectures	15, 19
via generalization	1, 21, 40, 64
via specialization	22, 60, 65
Performance	5, 11, 50
GPU / FPGA	16, 36
Data at Rest	26, 42
Data Model	52, 53
Data in Motion	13, 47
Applications	
Query Optimization	23, 66
Graph Queries	27, 28, 35, 62
Visualization	12, 59
Security	34, 70
Accuracy	51
Classification	18, 37
Decision Trees	7, 38, 68
Classifier Chains	20, 39
Nearest Neighbor	32, 56, 61
Ensemble Techniques	3, 31, 43
Decision Makers	2, 46
Clustering	58, 63
Social Media	49, 55
Stock Market	4
High Dimensional Data	10, 17, 54, 69

D. Data Extraction

We extracted the following throughput metrics:

- *Rate* R (bytes/second). The amount of data a system can process over the time necessary to process that data.
- *Data Size* D (bytes). The total amount of data processed.
- *Duration* t (seconds). The time it took to process data.
- *Sample Size* $\overline{S_i}$ (bytes). The average size of each sample.
- *Number of Samples* $|S|$. The number of samples.
- *Nodes* N (positive integer). The number of workers partaking in each operation.

Few studies directly reported the metric of primary interest to us: the rate at which data was processed (Table II). However, with the exception of N, these metrics can be defined in terms of one another (equation 1).

$$D = R \cdot t \qquad D = \overline{S_i} \cdot |S| \tag{1}$$

Therefore, we calculated values for unreported metrics for as many publications as possible. For all remaining publications, we used data imputation to infer values for metrics that were not reported directly and could not be calculated algebraically.

[1] https://ieeexplore.ieee.org
[2] https://dl.acm.org/
[3] https://scholar.google.com/

TABLE II
PUBLICATIONS REPORTING THROUGHPUT METRICS BY CATEGORY

Category	Reported (or Calculated)	Inferred	Insufficient Info
Nodes	55	0	15
Samples	45	5	20
Duration	31	5	34
Data Size	9	30	31
Rate	6	23	41

TABLE III
SIZE APPROXIMATIONS USED IN IMPUTATION

Classification	Attribute Type	Size (bytes)
Numeric	dimension, attribute, sample	4
(Numeric, Numeric)	point, location, graph edge.	8
String	tweet, word, log, trace, request.	140

E. Data Imputation

We employed statistical imputation to infer values for missing metrics. We assigned values to attributes based on their size and type (Table III). For numerical types, we assumed that the 32-bit representation (either a float or an integer) would be sufficient to propagate the necessary information through a distributed system. For types consisting of two points, we chose 8 bytes or the sum of two 4-byte attributes. For text types, we chose 140 bytes to mirror "tweet" [8].

We inferred data size from 30 publications and processing rate from 23 publications (Table II). However, imputation also introduced a threat to validity. Any imprecision in our selection of size approximations (Table III) propagated to the results. To mitigate these threats to the conclusion validity of this study, we present imputed results metrics as *Inferred*. Therefore, the reader can clearly identify which results are based on data reported directly versus results inferred via imputation.

III. RESULTS & DISCUSSION

A. G1 Findings

Prior research has focused on two main research topics: the design and architecture of distributed systems, and the applications that run on distributed systems. These two topics were clearly discernible from both the manually coded (Table I) and NLP (Fig. 1) results. These topics represent the two highest-level topics identified from the manual coding; the most common words identified via semi-automatic coding confirmed the importance of these two topics. The terms "System(s)" and "Application(s)" fell within the top 10 words and are the two most informative words in the top 10. Other words in the top 10 (e.g., "Stream(...)" and "Time") were topical but did not assist in discerning research topics whereas other words reflected terms commonly used in research publications and were neither topical nor discerning (e.g., "Based", "Two", and "Results").

Both manual and semi-automatic coding agreed on other frequently used terms. Examples of such terms include "Performance", "Network", "Architecture", and "Graph". The agreement in these terms demonstrated that both coding approaches detected similar patterns in the field of study. However, the results of the coding approaches were also complimentary. For instance, "Parallel" and "Mining" were only identified by the semi-automated approach whereas "Dynamic" and "Query" was only identified by manual coding.

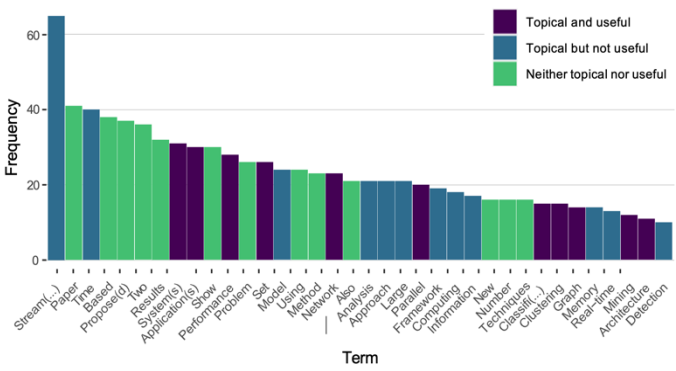

Figure 1. Frequency of Abstracts Containing the Terms Indicated on the X-axis. Bar color indicates whether the term was topical and useful, topical but not useful, or neither topical nor useful. Stemmed terms indicated by "(...)"; terms where singular and plural instances were combined indicated with "(s)".

B. G2 Findings

The field lacks standard and consistent metrics for reporting findings. Note the number of metrics in Table II that were neither reported nor could be calculated algebraically. A lack of comparable metrics between studies precluded efficient inter-study comparison and meta-analysis. It also hampered benchmarking, as imputation can introduce inconsistencies in results. Nevertheless, we characterized the throughput of distributed processing systems to the extent possible.

With respect to nodes (Fig. 2A), a large percentage (44%) of distributed system research was not conducted on physically distributed systems (i.e., research was conducted on one node on 31 of the 70 papers in our corpus). Further, only 6 papers reported researching systems larger than 16 nodes, with only 1 paper investigating a 144-node system.

With respect to the number of samples being processed (Fig. 2B), most publications reported numbers between one thousand and ten million. Only 14 publications processed data sets with 10 million samples or more. Three publications reported handling data sets on the order of billions of samples. Without additional context, it is difficult to interpret these results because samples can vary in size.

With respect to size (Fig. 2C), the paucity of data was surprising because knowledge of how much data was being processed is foundational for comparing experiments with one another. The largest data set was nine terabytes. Given that the average size of nodes was approximately 16 gigabytes, a distributed system is a promising avenue for data processing.

With respect to runtime (Fig. 2D), most publications reported experiment durations between 1 sec. and 16 min. Given that stream processing systems can be run continuously, these short experiments offered little inference for long-running experiments or the maintenance of distributed systems. These limited durations are concerning for professionals looking into using a distributed solution for their data processing needs because the paucity of information about the long-term utility

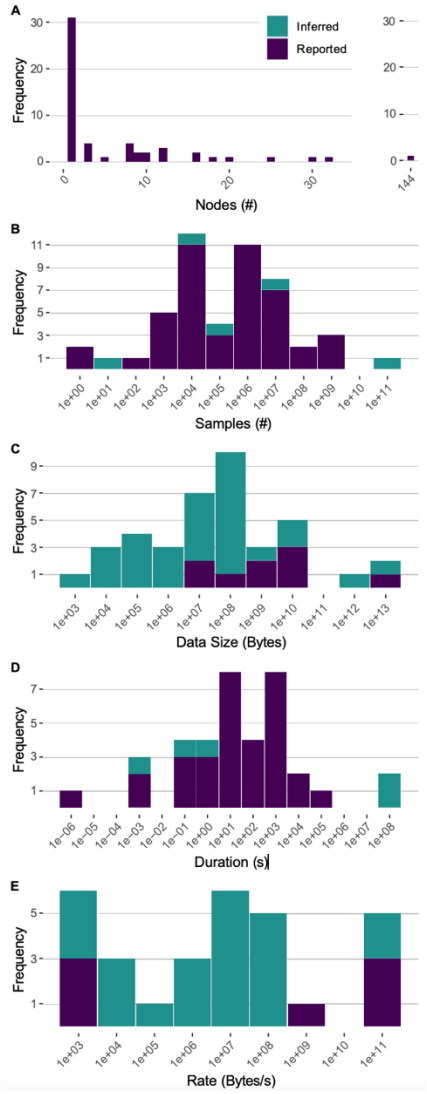

Figure 2. Histograms of Key Metrics. "Frequency" (y-axis) is the count of publications in the corresponding bars on the corresponding x-axis. Results directly reported or algebraically calculated are shown in dark purple; results inferred via imputation are shown in blue-green color.

and maintainability of distributed systems makes it difficult to form a basis for judgment.

With respect to rate (Fig. 2E), most were inferred via imputation due to insufficient reporting. Recall, only 7 of 70 publications reported the rate at which data could be processed by their distributed system (Table II). However, the rate is a critical metric and perhaps the most relevant metric for consumers of distributed processing systems.

C. G3 Findings

From the early 2000s to 2016, research in the area of distributed data processing increased (Fig. 3). Thereafter, research publication numbers fluctuated. Note that we formed

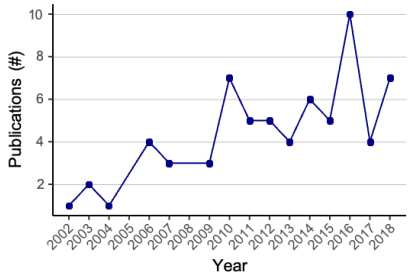

Figure 3. Count of Distributed Data Processing Publications in Corpus

our corpus of publications during 2019 and therefore did not have a complete count of publications for that year.

IV. Conclusion and Future Directions

Since the early 2000s, the field has grown, and he most active areas of research included algorithm design and system design. Many researchers are contributing to ever-increasingly specialized optimizations to be performed over the general pattern of distributed processing. Despite the growth, a standardized set of metrics for reporting results is lacking. Without such standards, the comparison of experimental results from one study to the next is hampered. A core set of metrics should be reported, including those listed in Table (II).

Importantly, the scope of academic research is limited in comparison to commercial applications. The scale that industry leaders such as Facebook, Amazon, Netflix, and Google operate is significantly greater than any study reviewed here. Future academic research on this topic would benefit from industry partnerships, simulation experiments, and exploring the use cases under which distributed data processing is—and is not—the most advantageous solution available. The corpus of new papers is also growing, and we intend to investigate publications through 2023.

References

[1] Akoka J., Comyn-Wattiau I., and Laoufi N. Research on big data–a systematic mapping study. *Comp. Stand & Interfaces*, 54:105–115, 2017.

[2] Fahed Alkhabbas, Romina Spalazzese, and Paul Davidsson. Characterizing internet of things systems through taxonomies: A systematic mapping study. *Internet of Things*, 7:100084, 2019.

[3] Sean Bleier. Natural language took kit's list of english stopwords. https://gist.github.com/sebleier/554280, 2021. Accessed: 2021-01-24.

[4] D. Budgen, M. Turner, P. Brereton, and B. A. Kitchenham. Using mapping studies in soft. engineering. In *PPIG*, V8, pg. 195–204, 2008.

[5] Omri M.N., Helali L. A survey of data center consolidation in cloud computing systems. doi.org/10.1016/j.cosrev.2021.100366, Feb 2021.

[6] Machine Learning Group University of WAIKATO. Weka 3 - data mining with open source machine learning software in java. https://www.cs.waikato.ac.nz/ml/weka/, 2021. Accessed: 2021-01-24.

[7] Thies W., and Amarasinghe S. An empirical characterization of stream programs and its implications for language and compiler design. In *2010 19th Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, pages 365–376. IEEE, 2010.

[8] Twitter. Decahose api. https://developer.twitter.com/en/docs/twitter-api/enterprise/decahose-api/overview/decahose, 2021.