Vrije Universiteit Amsterdam

Using NLP to conduct a study of selected performance metrics in graph computing and the operational techniques used to achieve them

Author: Paul Puettbach (2622550)

August 5, 2023

Contents

Abstract

Many real world problems from various research domains can be mapped well to graph analysis problems (e.g.[9]). Many of the corresponding input graphs are large enough to require distributed execution. For example there have been social graphs with billions of edges[3]. Improving the performance of distributed graph analysis solutions means improving the scale of the problems that can be solved, the speed at which they can be solved, and the resource consumption cost that is incurred by solving them. One obstacle in this endeavour is the lack of congregate information on the operational techniques most commonly used in relation to the optimization space. To alleviate this issue, this paper provides an overview taxonomy of graph processing solutions and the operational techniques associated, as well as the performance metric that is optimized using the operational technique. Furthermore, this paper introduces a novel data driven methodology for a Natural-Language-Processing (NLP) guided literature selection process to better ensure the breadth of research is taken into account during the analysis and construction of the taxonomy.

- (still missing is a brief sentence about the findings)

1 Introduction

1.1 Relevance

Massive distributed graph processing is an impactful research field. Many big data research and real world problems are naturally expressable as graph problems, such as protein interactions in bioinformatics [9]. Therefore, optimizing the approach to distributed graph processing positively impacts current research and aids solving real world problems. Improving the performance of graph processing solutions in breast cancer research such as [26] for example, will have a positive impact on the survivability of people with a breast cancer diagnosis.

1.2 Problem Statement

To aid the optimization of graph processing solutions it is important to gain an overview of the operational techniques used to achieve certain performance metrics for these solutions. A more qualitative analysis utilizing exemplary research and mapping its operational techniques to the performance metric that was optimized will give insight into the gaps in research and the status quo. The more a methodology can claim to give an overview over the full breadth of research, the more impactful the analysis of the body of research. Like other quantitative research the sampling process (paper selection) in a literature survey is vital since the more representative the sample (the selected papers) is of the general population the stronger the inferences made off of the sample in regards to the overall population (all papers in the research field). There exits a plethora of convectional structured literature survey techniques [17] but they suffer from human bias in the paper selection process, thereby weakening any claim made on the research field at large. Whether the selected papers are representative for the field at large or if they are relevant in the field is judged by the surveyor. This is inherently biased and weakens any extrapolation of findings to the field at large. Additionally, even if the selected paper of a conventional survey methodology are a relatively fair representation, within a research group the papers that seed a literature study are more stagnant which undermines variance in the research taken into account within one group. Hence, using the NLP data driven paper selection process outlined in this paper will strengthen any extrapolation made using the sample.

1.3 Contributions

This paper makes the following contributions:

- a taxonomy of the papers of energy consumption, latency, and scaleabilty mapped to operational techniques
- a novel methodology for the paper selection process with a codebase to replicate the methodology
- an evaluation of the taxonomy and contextualization of the selected papers in the research field at large.

1.4 Open Science

how to replicate the results link to the GitHub

1.5 Literature Survey Structure

The rest of the paper is organized as follows: **Background** will give background on operational techniques, NLP, and AIP. **Literature Survey Methodology** will explain the stages of the literature study. **Selected And Rejected Papers** will give insight into the research chosen for this meta analysis and **Results** will explain findings and evaluate the resulting taxonomy. Furthermore **Discusion** gives insight into the implications of the findings as well as the expierience with using this survey methodology. **Limitations** will elaborate on threads to validity and **Conclusion** will summerize the findings of this paper

2 Background

2.1 AIP

Article Information Parser is a instrument that combines three article datasets (DBLP [2], Semantic Scholar [4], and AMiner [1]) and "parse[s], unif[ies], and in some cases correct[s] article metadata. AIP creates a PostgreSQL database that allows for easily finding related work" [12]. The research available through AIP is multidisciplinary and diverse and has to be queryed using SQL to generate subdatasets of more relevance to the intended research field. AIP is used in this paper to generate these subdatasets of relevant papers for NLP processing.

2.2 Operational Techniques

The Operational Techniques are outlined in "Multivocal Survey of Operational Techniques for Serverless Computing"[30] with " Serverless Operational Techniques are the automated mechanisms that take place in the serverless platform to manage resources and enforce a certain level of QoS." These techniques are used in this paper to categorize the methods found to improve a given performance vector (Latency, Energy, Scalability). The following list quotes the descriptions for each Operational Technique found in [30].

- Replication: A technique for adding redundancy by creating multiple copies.
- Caching: A technique for low-latency lookups by placing replicas in proximity to the requester.
- Partitioning: A technique for splitting resources into smaller parts.
- Consolidation: A technique which combines multiple processes onto one resource.
- Migration: A technique for moving tasks to other resources.
- Offloading: A migration technique for out sourcing tasks to less constrained resources.
- Provisioning: A technique for acquiring new resources.
- Allocation: A technique for the mapping of tasks to resources. The mechanisim can already have resources acquired or will acquire them in the future.
- Elastic Scaling: A technique which automatically provisions or de-provisions resources.
- Load Balancing: A technique for spreading a load over multiple resources.

The operational techniques are not defined as clearly distinct and segregated. [30] outlines a more complex taxonomy for the operational techniques that often overlap or are subsets of each other.

2.3 LDA

Latent Dirichlet allocation (LDA) is Bayesian generative statistical model. It uses a form of unguided non deterministic NLP clustering algorithm that starts with random seed and iterates over each word in the corpus to constructs clusters called topics, "each topic is characterized by a distribution over words." [8] A words assignment to a topic is based on topic membership of identical words in the entire corpus and topic distribution of the words associated document. Each iteration of the algorithm a word is reevaluated to be assigned to the topic most fitting for it. Eventually the model will converge on a (semi) stable state. LDA is used to construct the topics for the queried corpus of considered literature in this paper.

Energy subdataset	
number of papers	10
sum of citations	67
max of citations	28
median of citations	3.0
mean of citations	6.7
subdataset specific SQL keywords	"energy", "green", "cooling", "heat", "resource utilization"

(a) Table 1

Scalability subdataset	
number of papers	47
sum of citations	376
max of citations	102
median of citations	3.0
mean of citations	8.0
subdataset specific SQL keywords	"scalability", "scale up", "extensibility"

(b) Table 3

Latency subdataset	
number of papers	34
sum of citations	648
max of citations	179
median of citations	4.5
mean of citations	19.058823529411764
subdataset specific SQL keywords	"latency", "runtime", "time complexity", "time-complexity", "TTC", "time to completion"

(c) Table 2

Figure 1: An overview of the generated subdatasets

3 Literature Survey Methodology

3.1 Generating The Subdataset

This paper evaluates three performance metrics for distributed graph processing solutions: energy consumption, latency, and scalability. For each a subdataset is queried from the AIP dataset, to be processed individually. The SQL queries to assemble the three subdatasets are comprised of common keywords used in each subdataset query such as: "graph processing", "serverless", "cloud", "cluster" etc. and keywords specific to the individual subdataset such as: "scalability", "scale up", "extensibility" for the scalability subdataset. A paper in the AIP dataset would need at least one keyword of each keyword set in the title or abstract to be eligible for the subdataset. Keyword selection is important since it dictates the intrinsic trade-off between the number of relevant papers not considered (false negatives) and the number of not relevant papers considered (false positives) of the subdatasets. To help alleviate this issue the proposed methodology will allow to redraw less relevant papers in the paper selection process. Therefore the keywords can be optimized to be more broad and decrease the number of false negatives in this step. The resulting three subdatasets are summerized in figure 1

3.2 Generating The NLP Model

To generate a quality NLP model the dataset has to be preproccessed. For the later model the abstract and title are taken into account so for each paper they are combined into a token set. Punctuation is removed and capital letters are converted to lowercase to avoid equivalent words in different places in the sentence resulting in different tokens. Numbers are removed from the token set as they should not be considered for the model since two numbers result in the same token while not being semantically similar. Bigrams and trigrams are added to the token set since they are sementically relevant only in conjuction to another. Lastly for the preprocessing step, stopwords are removed. In addtion to more traditional stopwords, stopwords for scientific writing (from, edu etc.) and the keywords of the query are taken into account. After the preprocessing step the model parameters are choosen. The number of topics will be the number of papers choosen for the subdataset one for each topic. Therefore, the number of topics is equivalent to the granularity of the literature survey. The number of passes and iterations are choosen to increase topic coherence and model quality and are determined by apriori trial runs of the model construction and evaluation of these trial models. LDA models are non deterministic and repeat computation may result in different if largely isomorphic models (different topics of different models tend toward being similar). To alleviate this problem, this paper constructs a number of models and compounds them into one using topic similarity and the property of near isomorphism amongst the models to match the most similar topics from each of the models. This Average Topic Model (ATM) strategy is outlined in [19]. Using the assumption that equivalent tokens contain similar semantic meaning, and that a paper is more likely to contain few topics, the resulting topics are a way to divide the full breadth of research into clusters of semantic meaning with each cluster being as dissimilar to each other cluster as the granularity chosen allows. Therefore, electing one representative paper per topic means a stronger claim to have considered the full breadth of the subdataset of research.

3.3 Paper Selection Process

Utilizing the ATM, a representative paper is picked from each topic. A paper is representative if it has the highest importance metric of all of the papers. The importance metric is the normalized sum of the topic adherence of each paper to the topic currently considered and the number of citations normalized with the sum of citations for that subdataset. The topic adherence accounts for 90 percent of the importance metric and the citations for 10 percent. The split between the two metrics is mutable and chosen based on an apriori dataset inspection using the mean median and maximum number of citations to assess the skeewdness of the citations. The more skeewed the number of citations the less the number of citations should account for in the importance metric. Once one candidate paper has been picked for each topic they are manually evaluated to eliminate false positives. If a paper is determined to be outside of the scope or interest of the literature survey they are redrawn choosing the paper with the next highest importance metric for the topic to replace the candidate paper. Once all candidate papers have been evaluated to be within the scope and interest of the literature survey they are marked as the chosen papers.

3.4 Mapping To The Taxonomy

The Overview of the Operational Techniques explained in [30] shows that some of the Operational Techniques overlap or completely encompass each other. Therefore mapping to the Operational Techniques is non trivial. Standerdized rules for that mapping are outlined in this section. If the operational technique used in the distributed graph processing solution is a subset of another operational technique only the subset technique will be marked. If the technique used lies at the intersection it will be mapped to both. If it used multiple methods each will be mapped. If the paper evaluated multiple platforms or methods each will be mapped. If the method is not described enough in the paper that an inference would have to be made or secondary literature consulted it will not be mapped. Partitioning is chosen if the decision is made based on the graph, and allocation if it is made based on properties of the node (such as left over compute capacity). If the improved performance of the graph processing solution stems from using non typical hardware, then in a heterogeneous cluster with standard and non typical hardware the performance would come from moving the tasks to the non standard hardware. Therefore, that graph processing solution would be mapped to migration. Additionally, Offloading and Consolidating are differentiated not by the direction of the word transfer (push or pull based) but by how the performance is gained. If the performance increase is the result of equalizing a workload imbalance it is Offloading if it is the result of sorting the type of tasks onto the nodes it is considered Consolidating. For example a node moving a vertex to another node that contains more of that nodes neighbours in local memory is Consolidating.

4 Selected And Rejected Papers

4.1 Selected Papers

For each perfromance metric of Energy consumption, Latency, and Scalability 8 papers were chosen one from each of the 8 topics generated by the ATM LDA model. Unfortunatly even with chosing the keywords such that we minimize the number of false negatives (papers that are elegible but not taken into account) only 10 papers were eligble to be considered and of those only 7 matched the scope and focus of this literature study.

The 7 selected papers for Energy consumption and their id label for the taxonomy are:

paper name	id
Massive Graph Procesing on Nanocomputers [24]	E01
A Case for Energy-Efficient Acceleration of Graph Problems using Embedded FPGA-based SoCs [21]	E02
Fine-grained power analysis of emerging graph processing workloads for cloud operations management [29]	E03
An elasticity study of distributed graph processing [6]	E04
Sparse Graph Processing with Soft-Processors [16]	E05
Optimizations and Analysis of BSP Graph Processing Models on Public Clouds [25]	E06
A comparative evaluation of open-source graph processing platforms [22]	E07

The 8 selected papers for Latency and their id label for the taxonomy are:

paper name	id
A Distributed Multi-GPU System for Fast Graph Processing [15]	L01
A Framework for FPGA Acceleration of Large Graph Problems: Graphlet Counting Case Study [7]	L02
Achieving up to Zero Communication Delay in BSP-based Graph Processing via Vertex Categorization [33]	L03
ExPregel: a new computational model for large-scale graph processing [27]	L04
FENNEL: Streaming Graph Partitioning for Massive Graph Scale Graphs [31]	L05
High-Performance Design of Apache Spark with RDMA and Its Benefits on Various Workloads [20]	L06
Start Late or Finish Early: A Distributed Graph Processing System with Redundancy Reduction [28]	L07
Towards Effective Partition Management for Large Graphs [32]	L08

The 8 selected papers for Scalability and their id label for the taxonomy are:

paper name	id
A Cost-Efficient Auto-Scaling Algorithm for Large-Scale Graph Processing in Cloud Environments with Heterogeneous Resource [13]	S01
CCFinder: using Spark to find clustering coefficient in big graphs [5]	S02
Concurrent Hybrid Breadth-First-Search on Distributed PowerGraph for Skewed Graphs [18]	S03
DISTINGER: A Distributed Graph Data Structure for Massive Dynamic Graph Processing [11]	S04
FBSGraph: Accelerating Asynchronous Graph Processing via Forward and Backward Sweeping [34]	S05
MaiterStore: A Hot-Aware, High-Performance Key-Value Store for Graph Processing [10]	S06
MESH: A Flexible Distributed Hypergraph Processing System [14]	S07
Scalable Big Graph Processing in MapReduce [23]	S08

4.2 Rejected Papers

A number of the candidate papers the model initially proposed had to be rejected due to them being unfit for the literature. A full list of the rejected candidate papers and the reason for the exclusion to this literature survey is provided in this section.

For the Energy cost performance metric the following 2 papers had to be rejected for the following reasons:

- Efficient graph computation on hybrid CPU and GPU systems: This paper was rejected because the graph processing solution proposed is non distributed which is outside of the scope of this literature survey.
- iGiraph: A Cost-Efficient Framework for Processing Large-Scale Graphs on Public Clouds: This paper decreases the monetary cost of large scale graph processing on public clouds, it does not decrease the energy consumption. Therefore, it was not choosen.

For the Latency performance metric the following 9 papers had to be rejected for the following reasons:

- Evaluation and Trade-offs of Graph Processing for Cloud Services: This paper was not chosen because it evaluates graphChi and other single node systems which are not distributed and therefore outside of the scope of this literature survey
- ATMoN: Adapting the "Temporality" in Large-Scale Dynamic Networks:This paper focuses on runtime knowledge based optimization it does not decrease runtime. Therefore, it is rejected
- Large-Scale BSP Graph Processing in Distributed Non-Volatile Memory: This paper optimizes for scalability not Latency and cannot be picked for the papers optimizing primarily for Latency
- Systems for Big-Graphs: This was part of the AIP database but it is not a paper it is a online tutorial and thereby the wrong format.
- Performance Characterization of Multi-threaded Graph Processing applications on Many-Integrated-Core Architecture: This paper contrast and compares various Many-integrated-Core processors placing it outside of the focus of this literature survey. The paper was selected by the model as a canidate paper because it eloborates on the low latency of the sub-NUMA cache clustering mode.
- PREDIcT: Towards Predicting the Runtime of Large Scale Iterative Analytics: The aim of this paper is to predict runtime not improve on it. Therefore there is no graph processing solution optimizing for latency to evaluate.
- Chaos: scale-out graph processing from secondary storage: Chaos is a graph processing solution that focuses on scalability not latency and can therefore not be considered for latency. It achieves higher scalabilty by load balancing at runtime which is why it contained the query keyword runtime.
- Performance Model for Parallel Matrix Multiplication with Dryad: Dataflow Graph Runtime: is not chosen since it adds a runtime overhead to model performance this graph processing solution adds runtime overhead to model performance. Analyzing the operational techniques used would be incorrect as it does not improve the latency overall.
- TrillionG: A Trillion-scale Synthetic Graph Generator using a Recursive Vector Model: TrillionG is a graph Generator it does not process the graph with a lower TTC and is therefore outside of the scope of this literature survey.

For the Scalability performance metric the following 1 papers had to be rejected for the following reasons:

- Declarative and distributed graph analytics with GRADOOP: This paper was rejected since it only demonstrates that the system is able to be scaled at all. The focus is not on optimizing scalability

In summery, keeping the query keywords broad to decrease the number of false negatives resulted in more work to seperate the canditate papers that could not be included in the literature survey. This work is necessary since it is the aim of the methodlogy of this literature survey to strengthen the claim of considering the full breadth of research. Some papers were rejected because they were outside of the scope of this literature survey and had been falsely queried by the shared keywords of the three subdatasets. Mutliple papers focused on non distributed systems for example and had been queried since they used keywords to contrast their approach with distributed systems. Some papers were falsely queried because the keywords were used in a different context than distributed graph processing such as hardware latency instead of the latency of the graph processing solution. It is important to note that while it is preferable to keep the number of false negatives low, an excessive number of false positives will impact the model quality. Despite, being able to filter them out as falsely considered candidate papers they were already considered as part of the LDA model and impacted its topic construction since it would be too costly to generate a new LDA model every time a canidate paper is rejected. However, if the number of rejected candidate papers is grossly disproportional to the total number of queried papers it would be necessary to reconfigure the keyword selection used in querying the AIP database and start anew at this step.

	Energy	Latency	Scalability
Replication		L02, L07	S03, S05, S07
Caching			S02, S05, S06
Partitioning	E06, E07	L01, L03, L04, L05, L08	S01, S02, S05, S07
Consolidation		L07	
Migration	E01, E02, E05	L01, L02, L06	S01
Offloading		L01	S06
Provisioning			
Allocation		L07	S03, S04, S05
Elastic Scaling	E03, E04		S01
Load Balancing	E03, E06	L01, L02, L05	S01

Figure 2: Taxonomy of Operational Techniques and Performance Metrics

5 Results

The taxonomy derived from mapping the operational techniques to the performance vectors is presented in Figure 2. This section discusses the results of this literature survey with a focus on the taxonomy and trends observed in the selected paper for each of the three performance vectors.

5.1 Energy

5.1.1 Taxonomy

None of the graph processing solutions that were part of the representative papers for Energy consumption discussed using replication or caching to achieve lower energy consumption. This is to be expected as replication and caching both introduce redundant work and resource usage. While higher resource utilization and a race to idle is beneficial to energy consumption if the workload stays the same, if overhead in the form of caching or replication is introduced energy is used up doing this redundant work. Similarly, Consolidation, Offloading and Migration generate an overhead and introduce wasted cycles and therefore wasted energy. Therefore, there are no graph processing solutions for the selected papers of the energy subdataset that utilize consolidation offloading or migration in the sense of moving tasks from one node to another. There is however, an abundance of papers using specialized hardware to achieve lower energy consumption costs which are mapped to migration as discussed in section 3.4 Mapping To The Taxonomy. Additionally, there are no graph processing solutions among the papers representative for the energy consumption subdatasets that utilize Allocation. However, "*[o]ne can consider load balancing a form of allocation*" [30] and load balancing is utilized by the representative papers.

5.1.2 Trends

Among all of the graph processing solutions considered for this performance vector a few overlapping trends stand out. First among them, the usage of specialized hardware more efficient at handling more specific problems managing more work with an equal or lower energy consumption. For example "A Case for Energy-Efficient Accelera- tion of Graph Problems using Embedded FPGA-based SoCs" [21] utilizes ARM System on Chip processors for their low energy consumption coupled with FPGA accellerators to build a graph processing cluster. "Massive Graph rocesing on Nanocomputers" [24] takes this a step further by not using specialized equipment but simple nanocomputers to build a graph processing cluster that takes significantly less power to run while still maintaining tolarable latency. Another trend observed is that many graph processing solutions make use of the fact that while tasks completed goes up near linerly with resource utilization power consumption does not. Therefore, keeping resource utilization high among all nodes leads to more processing done for less power consumed. Partitioning is used to split the workload into appropriate chunks to use the nodes to their capacity. This is often used in conjuction with load balancing to assign the approapritate chunck to the approapriate nodes such that they are utilized fully. This is the case in "Optimizations and Analysis of BSP Graph Processing Models on Public Clouds" [25] where only a subset of the vertices is considered at one time as to avoid having to scale up to more machines. These "swaths"of vertices are then load balanced to the requisitioned compute nodes to utilize them fully without the need requisition more machines that could not be fully utilized. Partioning is also often used in conjunction with elastic scaling as is the case in "Fine-grained power analysis of emerging graph processing workloads for cloud op-erations management" [29] where the requisitioned machines are kept as low as possbile through scaling down as soon as a workload decrease allows.

5.2 Latency

5.2.1 Taxonomy

Examining the taxonomy for the graph processing solutions that are representative of the Latency subdataset it is apparent there are more papers that utilize multiple operational techniques and that more operational techniques are represented in general compared to the graph processing solutions aiming to optimize Energy Consumption. A lot of the papers use Partitioning which is problem domain specific to graph processing. With many graph processing algorithms, the graph vertices are largely dependent on their neighbour vertices to communicate updates with them. An inconsiderate partitioning scheme could negatively impact performance by increasing the number of cross node communications by increasing the average number of of-node neighbour vertices. Thereofore, alot of the papers utilize more elaborate partitioning schemes out of necessity. Also of note, a lot of the graph processing solutions use migration and again all in the form of specilize hardware. In contrast, there are no graph processing solutions that focus on caching or elastic scaling. It is assumed with more papers considered in the literature survez example papers for these operational techniques would be found. Additionally, caching is handled well by standard graph processing platforms already so many papers would focus on improving on less exhaustively explored fields. Furthermore, the improvements in latency should not stem from a simple increase in the available computing resources so using elastic scaling to requistion new resources trivializes decreasing latency, which is why it is most likely not a focus of many papers. It is only a worthwile improvement if the latency can be decreased while the available resources remain static.

5.2.2 Trends

All of the Operational techniques marked as Migration for the papers of this section are utilizing migration in the form of special hardware. "A Distributed Multi-GPU System for Fast Graph Processing" [15] uses a GPU cluster for their bigger on chip memory when compared to CPU's, this allows a GPU cluster too keep more of the graph data in their on chip memory. This, coupled with the fact that graph processing workloads tend to be memory bandwidth bound in their performance allows for a GPU cluster to achieve lower latency. Simiarly, "A Framework for FPGA Acceleration of Large Graph Problems: Graphlet Counting Case Study" [7] optimizes latency with FPGA accelerators. FPGA based cluster have increased off chip memory access latency. Instead of circumenventing this issue an decreasing the off chip memory access the solution presented aims to decrease the latency of off memory chip access the FPGA's expierience by connecting them to a multitude of memory banks connected with a memory crossbar. Similarly, "High-Performance Design of Apache Spark with RDMA and Its Benefits on Various Workloads" [20] also uses special hardware to improve latency in this case a NIC that allows for Remote-Direct-Memory-Access (RDMA) on the various compute cores. RDMA allows a core to access another cores memory without the usual checks and balances enforced by the cores OS. In a non adverseria cluster such checks and balances are not necessarly needed and using RDMA can significantly cut down on communication latency as is presented in [20].

Another apparent trend is using Partitioning in conjunction with load balancing to improve performance. "A Distributed Multi-GPU System for Fast Graph Processing" [15] uses a dynamic graph repartitioning scheme that evaluates the current execution time of each partition and then the estimated cost and gain of a global or local repartioning. Then if the gain outweighs the cost the graph is repartitioned and load balanced onto the nodes anew. Another graph processing solution that uses partioning in conjunction with load balancing is "FENNEL: Streaming Graph Partitioning for Massive Graph Scale Graphs L05" [31]. Fennel aims to strike a balance between the communication overhead generated by scheduling neighbouring nodes onto different machines, and the load imbalance that is due to minimizing the number of edges going to non machine local vertices. The partitioning scheme does not just take locality into account here but considers load balancing as well.

In contrast, a multitude of the graph processing solutions representative for the latency subdataset only use partitioning to increase locality. "Achieving up to Zero Communication Delay in BSP-based Graph Processing via Vertex Categorization" [33] categorizes vertices as local if they only have neighbours on the same node. Moving some of the computations done on these nodes onto the superstep of BSP optimizes the execution pipeline which usually leaves CPU's underutilized during the superstep. In order to benefit more from this idea the graph needs to be Partitioned in a way such that there are as many local vertices as possible, to increase the locality. Another paper that aims to increase the number of local vertices is "ExPregel: a new computational model for large-scale graph processing" [27]. The paper first aggregates all necessary values from local vertices and calculates the local value as far as possible with just that before sending the updated values to of node neighbour vertices. More local vertices means less messages have to go through the network stack and more of the total computations can be done with just local values. Therefore, ExPregel utilizes a partitioning scheme that also maximizes locality.

5.3 Scalability

5.3.1 Taxonomy

The graph processing solutions representative for the Scalability subdataset are the most diverse in terms of what Operational Techniques they use and how many Techniques are used concurrently by a sigle graph processing solution. A lot of them employ replication owning to the fact that "[m]any graph algorithms tend to explore the structure of the graph while performing a relatively small amount of computations. This results in a higher ratio of data access to computation compared to mainstream scientific and engineering applications, and combined with poor locality leads to execution times dominated by memory latency." [7]. This memory latency bottleneck because more debilitating the bigger the graphs and the more nodes have to communicate. Replication cuts down on this scaled memory overhead in various ways. Another prevelant Operational Technique to optimize Scalability is Caching which is prevelant for the same reason as Replication. As with the other Perfromance vectors Partitioning is frequent in the taxonomy due to the characteristics of graph processing. As the size of the graph scales up inbalances in the workload and decreasing efficiency due to the lack of locality are more problematic to performance if a well thought through partitioning scheme does not alleviate these issues. All of the papers in this section use Allocation in the form of temporal task placement, placing the task on the right resource is less improtant than placing the task on a resource at the right time.

5.3.2 Trends

Among the graph processing solutions representative of the Scalability subdataset some of the ones using Partioning use it in conjunction with other Operational Techniques to improve scalability. "A Cost-Efficient Auto-Scaling Algorithm for Large-Scale Graph Processing in Cloud Environments with Heterogeneous Resources"[13] for example, analyzes the communication pattern of the nodes to repartitione the input graph and re load balance it onto the VM's based on the inferred placement of the VM's on the physical servers in a cloud datacenter to increase locality. The paper also employs elastic scaling to scale up and down the type and number of VM's used. Conversely, "MESH: A Flexible Distributed Hypergraph Processing System" [5] uses Partioning and replication independently of each other. MESH evaluates three partitioning algorithms: random, greedy, hybrid to determine which one is the most suited for hypergraph processing. MESH's Replication scheme is in line with the Replication seen in "Concurrent Hybrid Breadth-First-Search on Distributed PowerGraph for Skewed Graphs"[18] and "FBSGraph: Accelerating Asynchronous Graph Processing via Forward and Backward Sweeping" [34]. All of them replicate vertices that would otherwise accrue a higher communication cost detrimental to scalability.

-partioning S01 analysis of the communication pattern to repatitione graphs onto the vms based on the infered placement of the vms on the physical servers in a cloud datacenter. workload varies over some graph processing algorithms (if they converge there might be less work in later iterations for example) they scale up and down the pool of the vms and then partition the graph and balance it onto the new vms

- S02 uses new graph representation datastruture to more efficiently partione the graph - S07 evaluates multiple partioning algorithms such as random, greeedy, and hybrid

-replication: all three replicate vertices that would otherwise encour a higher communication cost which is not good for scalability at this level of graph scale the tradeoff between computation overhead and communication overhead has to be taken. in S05 for example they partionine their graph along individual paths with no intersecting edges between paths, and then do a foward and backward sweep vertex communication scheme if a vertex is part of two different paths it has has to be replicated since otherwise nodes with a path assigned to them each and a vertex on remote paths would have to communicate

- this partioning scheme also allows to get more from caching most recently used vertices the end of the foward sweep is the start of the backward sweep which is why S05 also is part of caching. so for this paper the partioning schemes is the main driver for perfromance and then and the type of partioning scheme influences the performance of the caching and replication

caching S06 main performance driver is the caching it uses a special type of vertex usage aware cache and a prefetch buffer before the main memory. make use of the skewed power law degree distributing in real graphs (small amount of vertices have a very high degree). so many degree vertices should stay in the cache.

-both s04 and S05 talk about allocation in the form of overlapping the communication and the computation. S03 talks about timing the switch from bottom down distributed breadth first search and bottom up breadth first search so allocation of the new tasks of bottom up breadth first search is what allows for a more scalable system

6 Discussion

- implication of findings

- challenges with placing papers challanges with literature survey method

7 Limitations

- the remaining machine bias - the dataset bias some methods - the model problem not guaranteed to get good model in reasonable time and non deterministic

8 Further Research

9 Conclusion

References

[1] Aminer research literature dataset. https://www.aminer.org. Accessed: 27.07.2023.

[2] Dblp research literature dataset. https://dblp.uni-trier.de. Accessed: 27.07.2023.

[3] Scaling apache giraph to a trillion edges. https://engineering.fb.com/2013/08/14/core-data/scaling-apache-giraph-to-a-trillion-edges/. Accessed: 19.07.2023.

[4] Semantic scholor research literature dataset. https://www.semanticscholar.org/product/api. Accessed: 27.07.2023.

[5] Mehdi Alemi, Hassan Haghighi, and Saeed Shahrivari. Ccfinder: using spark to find clustering coefficient in big graphs. *The Journal of Supercomputing*, 73:4683–4710, 2017.

[6] Sietse Au, Alexandru Uta, Alexey Ilyushkin, and Alexandru Iosup. An elasticity study of distributed graph processing. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 382–383. IEEE, 2018.

[7] Brahim Betkaoui, David B Thomas, Wayne Luk, and Natasa Przulj. A framework for fpga acceleration of large graph problems: Graphlet counting case study. In *2011 International Conference on Field-Programmable Technology*, pages 1–8. IEEE, 2011.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[9] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res*, 31(9):2443–2450, May 2003.

[10] Dong Chang, Yanfeng Zhang, and Ge Yu. Maiterstore: a hot-aware, high-performance key-value store for graph processing. In *International Conference on Database Systems for Advanced Applications*, pages 117–131. Springer, 2014.

[11] Guoyao Feng, Xiao Meng, and Khaled Ammar. Distinger: A distributed graph data structure for massive dynamic graph processing. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1814–1822. IEEE, 2015.

[12] et al. Gilles Magalhaes. Aip. https://github.com/atlarge-research/AIP/commits/master, 2023.

[13] Safiollah Heidari and Rajkumar Buyya. A cost-efficient auto-scaling algorithm for large-scale graph processing in cloud environments with heterogeneous resources. *IEEE Transactions on Software Engineering*, 47(8):1729–1741, 2019.

[14] Benjamin Heintz, Rankyung Hong, Shivangi Singh, Gaurav Khandelwal, Corey Tesdahl, and Abhishek Chandra. Mesh: A flexible distributed hypergraph processing system. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 12–22. IEEE, 2019.

[15] Zhihao Jia, Yongkee Kwon, Galen Shipman, Pat McCormick, Mattan Erez, and Alex Aiken. A distributed multi-gpu system for fast graph processing. *Proceedings of the VLDB Endowment*, 11(3):297–310, 2017.

[16] N. Kapre. Sparse graph processing with soft-processors. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 33–33, Los Alamitos, CA, USA, may 2015. IEEE Computer Society.

[17] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33, 08 2004.

[18] Zengxiang Li, Shen Ren, Sifei Lu, Jiachun Guo, Wentong Cai, Zheng Qin, and Rick Siow Mong Goh. Concurrent hybrid breadth-first-search on distributed powergraph for skewed graphs. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 160–169. IEEE, 2018.

[19] Nicolas Francisco Lopez Giraldo. *Assessing and reducing the impact of LDA's non-determinism in software engineering*. PhD thesis, UC Irvine, 2014.

[20] Xiaoyi Lu, Dipti Shankar, Shashank Gugnani, and Dhabaleswar K Panda. High-performance design of apache spark with rdma and its benefits on various workloads. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 253–262. IEEE, 2016.

[21] Pradeep Moorthy and Nachiket Kapre. A case for energy-efficient acceleration of graph problems using embedded fpga-based socs. 2015.

[22] Xiaohui Pan. A comparative evaluation of open-source graph processing platforms. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 325–330, 2016.

[23] Lu Qin, Jeffrey Xu Yu, Lijun Chang, Hong Cheng, Chengqi Zhang, and Xuemin Lin. Scalable big graph processing in mapreduce. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 827–838, 2014.

[24] Bryan Rainey and David F. Gleich. Massive graph processing on nanocomputers. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3326–3335, 2016.

[25] Mark Redekopp, Yogesh Simmhan, and Viktor K. Prasanna. Optimizations and analysis of bsp graph processing models on public clouds. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pages 203–214, 2013.

[26] SungMin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *CoRR*, abs/1711.05859, 2017.

[27] Masoud Sagharichian, Hassan Naderi, and M Haghjoo. Expregel: a new computational model for large-scale graph processing. *Concurrency and Computation: Practice and Experience*, 27(17):4954–4969, 2015.

[28] Shuang Song, Xu Liu, Qinzhe Wu, Andreas Gerstlauer, Tao Li, and Lizy K John. Start late or finish early: A distributed graph processing system with redundancy reduction. *arXiv preprint arXiv:1805.12305*, 2018.

[29] Shuang Song, Xinnian Zheng, Andreas Gerstlauer, and Lizy K John. Fine-grained power analysis of emerging graph processing workloads for cloud operations management. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2121–2126. IEEE, 2016.

[30] Alexandru Iosup Stijn Meijerink, Erwin van Eyk. Multivocal survey of operational techniques for serverless computing. *Amsterdam, Netherlands, Vrije Universiteit.*, 02 2021.

[31] Charalampos Tsourakakis, Christos Gkantsidis, Bozidar Radunovic, and Milan Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 333–342, 2014.

[32] Shengqi Yang, Xifeng Yan, Bo Zong, and Arijit Khan. Towards effective partition management for large graphs. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 517–528, 2012.

[33] Xuhong Zhang, Ruijun Wang, Xunchao Chen, Jun Wang, Tyler Lukasiewicz, and Dezhi Han. Achieving up to zero communication delay in bsp-based graph processing via vertex categorization. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 112–121. IEEE, 2015.

[34] Yu Zhang, Xiaofei Liao, Hai Jin, Lin Gu, and Bing Bing Zhou. Fbsgraph: Accelerating asynchronous graph processing via forward and backward sweeping. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):895–907, 2017.