IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 4, 2020, accepted November 13, 2020, date of publication December 1, 2020, date of current version December 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041670

UnifyDR: A Generic Framework for Unifying Data and Replica Placement

ANKITA ATREY[1], GREGORY VAN SEGHBROECK[1], HIGINIO MORA[2], BRUNO VOLCKAERT[1], (Member, IEEE), AND FILIP DE TURCK[1], (Senior Member, IEEE)

[1]Internet Technology and Data Science Lab (IDLAB), Ghent University, imec, 9052 Ghent, Belgium
[2]Department of Computer Science Technology and Computation, University of Alicante, 03690 Alicante, Spain

Corresponding author: Ankita Atrey (ankita.atrey@ugent.be)

This work was supported by the Agentschap Innoveren & Ondernemen (VLAIO) Strategic Fundamental Research (SBO) under Grant 150038 (DiSSeCt).

ABSTRACT The advent of (big) data management applications operating at Cloud scale has led to extensive research on the data placement problem. The key objective of data placement is to obtain a partitioning (possibly allowing for replicas) of a set of data-items into distributed nodes that minimizes the overall network communication cost. Although replication is intrinsic to data placement, it has seldom been studied in *combination* with the latter. On the contrary, most of the existing solutions treat them as two independent problems, and employ a two-phase approach: (1) data placement, followed by (2) replica placement. We address this by proposing a new paradigm, *CDR*, with the objective of *combining data and replica placement* as a single joint optimization problem. Specifically, we study two variants of the CDR problem: (1) *CDR-Single*, where the objective is to minimize the communication cost alone, and (2) *CDR-Multi*, which performs a multi-objective optimization to also minimize traffic and storage costs. To unify data and replica placement, we propose a generic framework called *UnifyDR*, which leverages overlapping correlation clustering to assign a data-item to multiple nodes, thereby facilitating data and replica placement to be performed jointly. We establish the generic nature of UnifyDR by portraying its ability to address the CDR problem in two real-world use-cases, that of join-intensive online analytical processing (OLAP) queries and a location-based online social network (OSN) service. The effectiveness and scalability of UnifyDR are showcased by experiments performed on data generated using the TPC-DS benchmark and a trace of the Gowalla OSN for the OLAP queries and OSN service use-case, respectively. Empirically, the presented approach obtains an improvement of approximately 35% in terms of the evaluated metrics and a speed-up of 8 times in comparison to state-of-the-art techniques.

INDEX TERMS Data placement, replica placement, OLAP, online social networks, join-intensive queries, location-based services, scalability, overlapping clustering.

I. MOTIVATION

We live in an *information age*, where almost every day-to-day need of an individual is fulfilled by digitally enabled services. This digital revolution has led to an exponential increase in the scale of data, and today, many Internet-based services (e.g., Facebook, Netflix, *etc.*) offer data at a never-before-seen scale [10], [23]. Although advancements in enabling technologies such as *big data* and *cloud computing* have provided us with the necessary machinery and systems (e.g., Apache Hadoop [39] and Spark [46]) to perform data management at scale, effective strategies for data placement and partitioning remain crucial for ensuring the performance of such systems [15]. Having said that, the field of data placement has witnessed a humongous amount of research over the past two decades [3], [4], [17], [30], [36], [41], [43], [45], [48], [49].

To better motivate the need for scalable solutions to the data placement problem, we consider two popular application domains—(1) *Online analytical processing (OLAP)*, and (2) *Online social networks (OSNs)*. OLAP is a computing paradigm for exploration and knowledge discovery from large data warehouses, thereby being a cornerstone for business intelligence and analytics [8]. Since data warehouses

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu.

are usually stored in a distributed manner across multiple nodes, successful execution of OLAP queries requires inter-node transfer of database tables. It is intuitive that identifying a placement of database tables in distributed nodes to reduce the inter-node table transfer during query execution reduces to an instance of the data placement problem. Moving on to the second application domain, OSN services are the most popular Internet-based services in today's world [43]. While Facebook and WhatsApp are used by individuals to communicate with their friends across the globe, Twitter has become the most preferred channel for information dissemination such as traffic news, emergency services, *etc.*, to a large audience. Owing to the usage of OSN services at a global scale, the data of OSN users is usually stored in geographically distributed nodes. Thus, if a user wants to mention or access the profile of one of her friends, the profile specific data of the latter has to be transferred to a node closest (in geographical distance) to the former. To this end, even for the OSN use-case, identifying a placement of user data to minimize the inter-node migrations triggered from profile visits or user mentions reduces to an instance of the data placement problem. Note that in both aforementioned applications, *replication* is required to ensure fault tolerance, while also facilitating reduction in communication cost.

It is evident from the above discussion that both data and replica placement are important for scalable data management. Besides, replication or replica placement is intrinsic to data placement, which is substantiated by a cautious examination of the generalized data placement problem [17] and its objectives. What is more, replication oblivious data placement is a specific instance of the generic data placement problem. Therefore, both data and replica placement should be considered as objectives of a single joint optimization problem. Having said that, although the field of data placement has witnessed significant advancements [3], [17], [43], to the best of our knowledge, none of the existing techniques possess the capability of performing data and replica placement jointly. On the contrary, most of the existing techniques treat the two placement steps as independent problems, and perform data placement followed by replica placement (Fig. 1). A key limitation of this *ad hoc two-phase approach* is that it results in solutions of inferior quality.

To address the aforementioned limitation, in this article, we propose a unified paradigm of combining data and replica placement, called *CDR*, as a joint optimization problem. Motivated by the aforementioned applications, we study two variants of the CDR problem. Specifically, as discussed previously, the goal of data placement for the OLAP use-case is to minimize the inter-node database table migration during the execution of OLAP queries. Since the optimization is concerned with minimizing a *single objective*, i.e., the communication cost, we formally denote the problem as *CDR-Single*. Recall that since OSN services usually operate at a global scale, a placement that minimizes the communication cost alone by minimizing the inter-node data migrations

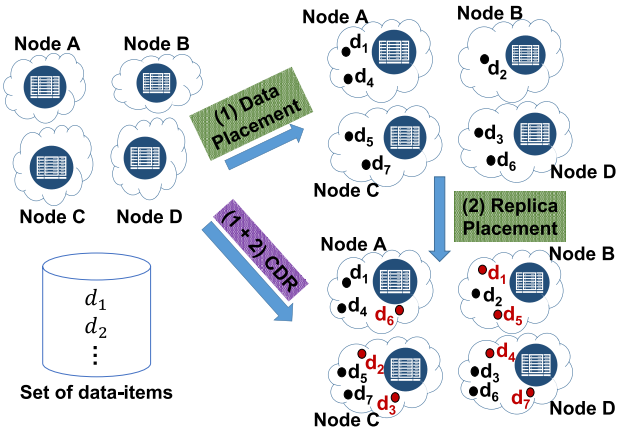

FIGURE 1. The standard two-phase data placement process (in green): where the data-items (black dots) are first placed in nodes and then replicated (red dots); and the proposed CDR paradigm (in magenta): where data and replica placement are jointly performed in a single step.

cannot be deemed as optimal. More specifically, since both user data and nodes are geographically distributed, factors such as inter-node latency, outgoing traffic, and storage costs, all of which are significantly different[1] for different geographically distributed nodes, need to be included in the optimization objective. Thus, we study a multi-objective optimization problem in the context of combined data and replica placement for OSN services, which is formally referred to as *CDR-Multi*. To solve both variants of the CDR problem, we propose a generic and unified framework, called UnifyDR, which leverages overlapping correlation clustering to address data and replica placement as a joint optimization problem. More specifically, overlapping clustering facilitates joint optimization of *data and replica placement in a single step* by allowing each data-item to be assigned to multiple nodes.

To summarize, we have comprehensively extended our previous work on combined data and replica placement [4] as a unified framework called UnifyDR. In addition to addressing the CDR problem for data-intensive OSN services in geographically distributed clouds (CDR-Multi) using overlapping clustering on hypergraphs, we solve the CDR problem for workflows originating in business analytics and intelligence, that of OLAP queries (CDR-Single), using graph-based overlapping clustering. This portrays the generalization ability of UnifyDR in addressing the CDR problem for a wide-variety of workflows originating from different real-world use-cases. Specifically, in contrast to [4], the following novel contributions and extensions are added in this work: (1) a new variant of the generic CDR problem, i.e., CDR-Single (Sec. III-B), (2) a generic framework UnifyDR to address the CDR problem under diverse settings (Sec. IV), (3)

[1]Note that in the majority of the cases OLAP queries are performed on data warehouses that are distributed on a cluster of (local) nodes, thus, factors such as inter-node latency, outgoing traffic, and storage costs are similar for different nodes, and hence, it is not required to include these factors in the optimization process.

a new algorithm for solving CDR-Single using overlapping clustering on graphs (Algorithm 1 and Sec. V), and (4) new empirical evaluations on data generated using the TPC-DS benchmark (Sec. VI-A).

Contributions.

- The study of a novel paradigm of combined data and replica placement, *CDR*, with the objective of *unifying* the aforementioned placement tasks into a single joint optimization problem (Sec. III). Motivated by two different real-world use-cases of OLAP and OSN services respectively, we study two variants of the CDR problem, called *CDR-Single* and *CDR-Multi*.
- A generic and unified framework, UnifyDR (Sec. IV), capable of solving the CDR problem in a single unified step as opposed to the traditional two-step process practiced by the existing state-of-the-art methods. UnifyDR also provides the ability to solve the CDR problem for workflows generated from different real-world applications, which is portrayed in this article using OLAP (CDR-Single) and OSN (CDR-Multi) use-cases.
- A novel algorithm based on *overlapping correlation clustering*, which allows each data-item to be assigned to multiple nodes (Sec. V). The proposed algorithm performs a single optimization for the OLAP use-case using overlapping clustering on graphs, while overlapping clustering on hypergraphs is employed for the OSN use-case to perform a multi-objective optimization for minimizing latency, node span, inter-node traffic, and storage cost.
- An extensive experimental study on data simulated based on the TPC-DS decision support benchmark and a trace of the Gowalla social network (Sec. VI), to showcase the effectiveness and scalability of the proposed UnifyDR framework and its associated CDR placement algorithm in solving the CDR-Single and CDR-Multi problems.

II. RELATED WORK

In the past decade, the data placement problem has witnessed extensive research with a wide variety of techniques developed for different execution environments, namely – distributed computing [9], [17], grid computing [13], [26], [27], and cloud computing [16], [19], [30], [44]. Initially, the focus of these works was on relational workloads such as database joins [17] and scientific workloads [14], [31], [45], however, recently the focus has shifted towards workloads emanating from specialized applications such as OSN services [21], [24] and data intensive services in geo-distributed clouds [1], [41]–[43], [47]. Given that our focus in this work is to combine data and replica placement as a single joint optimization problem, we only present a review of the existing literature on data placement that is directly related to our work.

The two main capabilities required to address the geo-distributed data placement problem are to capture and improve (1) data-item – data-item associations and (2) data-item – node associations. The former is measured as the frequency of co-occurrence of two or more data-items, whereas the latter is calculated using the frequency of occurrence of a data-item at a given node. While data-item – data-item associations were captured by methods relying on hierarchical clustering of data-item correlations [48], [49] and frequent pattern mining [33], literature also witnessed techniques [1], [22], [35], [47] capable of capturing data-item – node associations. Volley [1], a system proposed by Agarwal *et al.*, performs automatic data placement in geo-distributed nodes based on co-occurrence information mined from the server logs of node requests. Rochman *et al.* [35] design algorithms that not only are cost efficient but also capable of serving a significant portion of user requests raised within the same region, along with the ability to manage the dynamic behavior of user requests. It is important to simultaneously honor the node storage capacities as well as minimize the data communication costs. To this end, Zhang *et al.* [47] propose an algorithm based on integer programming.

Literature also includes works focusing on other aspects related to the geographically distributed data placement problem, such as development of specialized strategies for replication and data placement in multi-clouds. Shankarnarayanan *et al.* [37] proposed strategies for location-aware replica placement[2] in order to minimize inter-node communication costs and other node location specific metrics. Shifting the focus of our discussion to multi-cloud environments, Jiao *et al.* [24] present a technique that takes multiple optimization objectives, such as inter-cloud traffic and carbon footprint, into consideration to perform data placement in multi-clouds. Later, Han *et al.* [21] proposed an algorithm for OSN service data migration, which can adapt to the variation in data traffic in multi-clouds.

Having said that, none of the aforementioned techniques are capable of modeling both data-item – node and data-item – data-item associations.

Recently, methods based on hypergraph representation of data-items and nodes have been extensively used in the literature for data placement in geographically distributed clouds. Yu and Pan [41]–[43] introduce the use of hypergraph modeling and leverage a partitioning tool called PaToH [7] to design data placement algorithms for data intensive services. On the one hand, hypergraph modeling helps to simultaneously capture both data-item – node and data-item – data-item associations. On the other hand, publicly available specialized heuristics for hypergraph partitioning [7] enable graceful scaling of the aforementioned methods to large datasets. Moving further, Atrey *et al.* [3], [5] proposed an algorithm based on spectral clustering of hypergraphs, which portrayed quality similar to the algorithms proposed in [43], however, achieved superior efficiency and scalability owing to the use of randomized eigendecomposition techniques for factorizing the hypergraph laplacian.

[2]Please see [18] for an in-depth survey of replica placement algorithms.

Owing to their ability to capture both data-item – node and data-item – data-item associations, the methods proposed by Atrey *et al.* [3] and Yu and Pan [43] constitute the representative state-of-the-art for geo-distributed data placement of data-intensive services, and are therefore considered for empirical comparisons with UnifyDR in the OSN use-case.

(Hyper)graph based solutions have also been popular for data placement of more traditional workloads such as scientific and relational workflows. The existence of a polynomial-time reduction of the data placement problem into an instance of the graph partitioning problem was proved in [17]. Furthermore, Golab *et al.* [17] proposed three algorithms—an optimal algorithm based on integer linear programming (ILP) and two heuristics for practically solving the problem on large scale workloads—to perform data placement for join-intensive database queries and data-intensive scientific workflows by minimizing the overall network communication cost. With the objective of reducing the partitioning overhead, Quamar *et al.* [34] present SWORD, a light-weight and scalable approach for data placement of Online Transaction Processing (OLTP) workloads. Specifically, SWORD performs data placement in two phases. Hypergraph modeling and hash partitioning based compression of the constructed hypergraph are carried out in the first phase, which is followed by partitioning of the compressed hypergraph in the second phase.

Owing to their scalability and effectiveness, the techniques, Metis and Metis+H2, proposed by Golab *et al.* [17] constitute the representative state-of-the-art for data placement of traditional data-intensive workloads, and are therefore considered for empirical comparisons with UnifyDR in the OLAP use-case.

Based on the aforementioned discussion, all the existing state-of-the-art techniques lack the capability of jointly solving data placement and replication. On the contrary, these techniques treat the two placement steps as independent problems, and perform data placement followed by replica placement, thereby resulting in solutions of inferior quality. To the best of the authors' knowledge, the research presented in this article is the first effort towards combining data and replica placement (CDR) into a single joint optimization problem, which is solved using overlapping correlation clustering on graphs (for the OLAP use-case) and hypergraphs (for the OSN use-case). More specifically, overlapping clustering facilitates joint optimization of *data and replica placement in a single step* by allowing each data-item to be assigned to multiple nodes. To summarize, the UnifyDR framework provides an elegant solution to both variants of the CDR problem, i.e., CDR-Single and CDR-Multi.

III. COMBINED DATA AND REPLICA PLACEMENT

Although the CDR paradigm is generically applicable to myriad settings, we motivate CDR-Single specifically through OLAP join queries and CDR-Multi through location-based OSN services. We first introduce the basic terminology related to data and replica placement.

A. PRELIMINARIES

Definition 1 (Data-Items ($\mathcal{D}$)): A data-item is defined as an atomic unit of data storage and transfer. $\mathcal{D}$ denotes the set of data-items, where $|\mathcal{D}| = n$.

Definition 2 (Nodes ($\mathcal{N}$)): A node constitutes a set of resources to store the data-items and perform different computational tasks on the stored data-items. Nodes are denoted using the set $\mathcal{N}$, where $|\mathcal{N}| = l$.

It is common for data-items to be placed across geographically distributed nodes in large-scale systems. Naturally, migration of some data-items may be required for proper execution of various tasks. Having said that, a data-request pattern is comprised of data-items that require migrations. Formally,

Definition 3 (Data-Request Patterns ($\mathcal{R}$)): A data-request pattern $R \in \mathcal{R}$ is comprised of a set of data-items $D \subseteq \mathcal{D}$ that are required to be present together in a single node N_j for a given task to be executed. The data-items ($d_i \in D$) that are not stored in N_j are transferred from the nodes in which they are stored to N_j. The set of data-request patterns denoted as $\mathcal{R}$ represent the system workload.

In addition to distributing data across nodes, real-world systems usually store multiple replicas of each data-item as well. This is because replication helps in ensuring fault-tolerance, while also facilitating reduction in communication cost and retrieval latency by potentially allowing for data-item retrieval from a geographically closer node.

Definition 4 (Replication Factor (r)): The replication factor r is defined as the number of replicas stored for each data-item.

Given the replication factor, a set of nodes, data-items, and data-request patterns as input, the objective of CDR is to partition the set of data-items, allowing for replication wherever applicable, across distributed nodes in order to minimize the overall communication cost emanating from migration/replication[3] of data-items corresponding to different data-requests. At this juncture, we would like to clarify that the CDR placement algorithm presented in this work considers the system workload to be static. More specifically, any change (small or large) in the system workload would require re-execution of the full pipeline to obtain the placement output. This design decision is in line with almost every existent technique [3]–[5], [16], [17], [43], [49] in the extensive literature on data placement. Thus, making the CDR placement algorithm dynamically adapt to the changes in the system workload is not in the scope of the current work.

Having defined the basics, we next discuss concepts specific to OLAP join queries and location-based OSN services,

[3] Among others, packet loss and data-item retrieval delays are some of the additional overheads that might effect migration or replication of data-items. However, in this article we restrict our attention towards minimization of communication cost alone. Note that this assumption is only introduced for the sake of brevity, and is not limiting with respect to the capabilities of the proposed CDR placement algorithm, which remains generically applicable in a variety of settings.

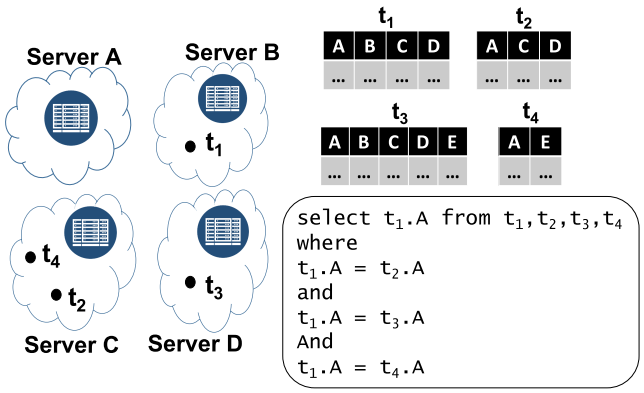

FIGURE 2. An example of OLAP join query.

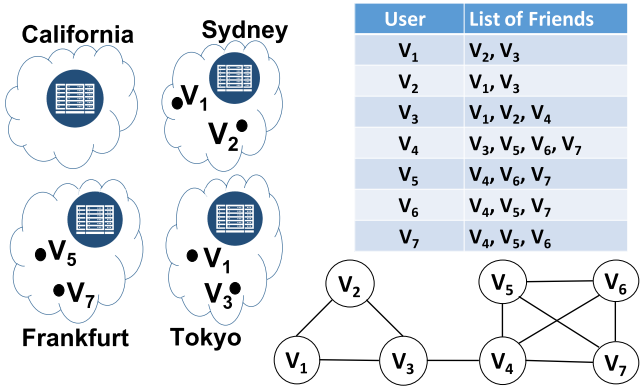

FIGURE 3. A location-based OSN service.

portray their relationship to the CDR problem, and formally define the CDR-Single and CDR-Multi problems.

B. CDR-SINGLE

As discussed in Section I, many OLAP queries involve database joins. A sample join query on a database comprised of four tables partitioned across four servers is portrayed in Fig. 2. There are two central aspects pertaining to OLAP queries: (1) a *database of tables*, where each table contains information specific to a real-world entity, and (2) an execution engine that allows users to submit and execute analytical *queries*.

Definition 5 (Database ($D(T)$)): A database $D(T)$ is a collection of information tables $T : |T| = n$, where each table consists of one or more attributes.

In the context of the OLAP use-case, each database table corresponds to a data-item (Def. 1). Thus, the set $\mathcal{D}$ contains n data-items corresponding to each table $t \in T$ of the database, where the data-item corresponding to a table t is denoted as $d(t)$.

Moving ahead, queries allow end-users to perform a variety of analytical tasks, such as computing the average quarterly sales and profit per product category. Such queries might require information from multiple database tables (often partitioned across servers), thereby triggering a data-request pattern (Def. 3) comprised of the tables that need to be migrated for proper query execution. To enable efficient query execution, naturally, each query is executed at a node $N_j \in \mathcal{N}$ (Def. 2) which requires the least number of tables to be migrated.

Thus, the data-request pattern $R(Q_k) \in \mathcal{R}$ triggered by a query Q_k at a node N_j is mathematically denoted as $\{d(t) \mid t \in \mathbf{QS}(Q_k) \wedge t \notin N_j\}$, where $\mathbf{QS}(Q_k)$ denotes the set of tables required for executing the query Q_k. Given this information, we formally define a query as follows:

Definition 6 (Queries ($\mathcal{Q}$)): A query $Q_k \in \mathcal{Q} \mid 1 \leq k \leq \eta$ signifies a request for the data-items contained in $R(Q_k)$ triggered from a node N_j capable of serving user requests. The set $\mathcal{Q}$ contains η queries and represents the system workload.

For example, the sample query Q_1 portrayed in Fig. 2 would be executed at the node N_3 (i.e., Server C), and would trigger a data-request pattern $R(Q_1)$ where $R(Q_1) = \{d(t_1), d(t_3)\}$.

Next, we provide a formal description of the CDR-Single problem, which is stated as follows.

Problem (CDR-Single): Given a set of n data-items $\mathcal{D}$ corresponding to the set of database tables T, η user queries $Q_k \in \mathcal{Q}$ representing the system workload, where each query comprises a data-request pattern $R(Q_k)$ being originated from a node $N_j \in \mathcal{N}$, and the replication factor r, perform combined data and replica placement to minimize the average number of nodes spanned $\mathcal{S}(R(Q_k))$ by the data-items corresponding to the request pattern $R(Q_k)$ of each query.

C. CDR-MULTI

As discussed in Section I, we study the CDR-Multi problem in the context of location based OSNs. There are two central aspects pertaining to location based OSN services: (1) a social network of users with network connections indicating friend relationships, and (2) a list of *check-ins* triggered by the users of the OSN service visiting diverse locations across the globe. A sample social network of seven users with six check-ins registered at four different node locations is presented in Fig. 3.

Definition 7 (Social Network ($G(V, E)$)): A social network with n individuals and m social ties can be denoted as a graph $G(V, E)$, where V is the set of vertices representing the users of the social network, $|V| = n$, and E is the set of edges (representing friend relationships) between any two vertices, $E \subseteq V \times V$, $|E| = m$.

For the OSN use-case, a data-item (Def. 1) corresponds to the most recent snapshot of a user's profile (e.g., comments, posts, profile picture, *etc.*). The data-item corresponding to a social network user $v \in V$ is denoted as $d(v)$, and there exists a total of n data-items (one for each social network user) in the set $\mathcal{D}$.

Moving further, check-ins characterize the OSN users' behavior of visiting different places in the world. A user check-in usually consists of two parts: (1) a geographic location in the world where the user registered the check-in, and

(2) a data-request pattern triggered in response to the user check-in. Note that the location recorded for a user check-in may be different from the location where the check-in was registered. This is because for each check-in, the recorded location is that of a node (Def. 2), which is closest (in geographical distance) to the location where the user registered the check-in. Thus, for OSN services, each node $N_j \in \mathcal{N}$ possesses a location attribute $N_j.loc$. The node locations are represented using the set $\mathcal{L}$, where $L_j = N_j.loc : N_j \in \mathcal{N}$, resulting in a total of $|\mathcal{L}| = l$ node locations.

Moreover, the data-request pattern $R(v) \in \mathcal{R}$ (Def. 3) triggered by a check-in from a user v at a node N_j is composed of the data-items corresponding to each of her friends. This is because a user may want to tag/mention some of her friends while registering a check-in. Mathematically, $R(v) = \{d(u) \mid u \in \mathbf{Adj}(v)\}$, where $\mathbf{Adj}(v)$ is the set of all the friends of the user v. Next, we provide a formal description of check-ins, which is stated as follows.

Definition 8 (Check-Ins ($\mathcal{C}$)): A check-in is a tuple $\forall k_{1\leq k\leq \rho}$, $C_k = (R(v), L_j) \in \mathcal{C}$ consisting of a data-request pattern $R(v) \in \mathcal{R}$ triggered by v and a location $L_j \in \mathcal{L}$ of a node N_j capable of serving user requests. The set $\mathcal{C}$ contains ρ user check-ins.

In other words, the check-in $C_k = (R(v), L_j)$ by a user v at a location L_j signifies a request for the data-items contained in $R(v)$ triggered from the node N_j located at $L_j = N_j.loc$. Considering the example presented in Fig. 3, if the first check-in was registered by the OSN user v_5 at the node N_3 with $L_3 = N_3.loc =$ Frankfurt, then $C_1 = (R(v_5), L_3)$, where $R(v_5) = \{d(v_4), d(v_6), d(v_7)\}$.

It should be noted that each check-in, even if it is registered by the same OSN user at the same location, is considered as unique or different from all other check-ins. This is required for modeling data-item – node and data-item – data-item associations appropriately. For instance, let's consider that the OSN user v_7 visited Frankfurt and Sydney 7 and 2 times, respectively. Intuitively, the association of the data-items contained in the data-request pattern $R(v_7)$ is relatively stronger with the node located in Frankfurt when compared to the one located in Sydney. This behavior is properly modeled by registering 7 different check-ins numbered $C_k, \ldots, C_{k+6}$ for the OSN user v_7 with the data-request pattern $R(v_7)$ at N_3 with $L_3 = N_3.loc =$ Frankfurt. Similarly, 2 different check-ins C_{k+7}, C_{k+8} for the same user v_7 are registered at N_2 with $L_2 = N_2.loc =$ Sydney. In the same vein, the data-item – data-item association between two data-items that co-occur more frequently (say, five times for data-items $d(v_3)$ and $d(v_4)$) in data-request patterns would be stronger when compared to that of data-items that are requested together rarely (say, only once for data-items $d(v_4)$ and $d(v_5)$). Furthermore, the aforementioned discussion provides substantive evidence in favor of our design choice to not index user check-ins uniquely using data-request patterns R and check-in locations L_j.

Having discussed the concepts specific to OSN services, the CDR-Multi problem is formally defined as follows.

Problem (CDR-Multi): Given a set of n data-items $\mathcal{D}$ corresponding to the set of social network users V, ρ user check-ins $C_k = (R(v), L_j) \in \mathcal{C} \mid v \in V$, $N_j \in \mathcal{N}$ representing the system workload, where each check-in comprises a data-request pattern $R(v)$ being originated from a node located at L_j, a set $\mathcal{N}$ of l nodes, with the per unit cost of outgoing traffic from each node $\Gamma(N_j) \mid N_j \in \mathcal{N}$, the per unit storage cost of each node $\Delta(N_j) \mid N_j \in \mathcal{N}$, the inter node latency (directed) for each pair of nodes $\kappa(N_j, N_{j'}) \mid N_j, N_{j'} \in \mathcal{N}$, the average number of nodes spanned by the data-items corresponding to each request pattern $R(v)$ being $\mathcal{S}(R(v))$, and the replication factor r, perform combined data and replica placement to minimize the optimization objective $\mathcal{O}$, which is defined as the weighted average[4] of $\Gamma(\cdot)$, $\kappa(\cdot,\cdot)$, $\Delta(\cdot)$, and $\mathcal{S}(\cdot)$.

IV. UnifyDR

In this section, we provide a description of the UnifyDR framework, its core components, and the underlying combined data and replica placement algorithm. An architectural overview of the proposed UnifyDR framework is presented in Fig. 4.

We begin by providing a description of the building blocks of the UnifyDR framework.

- **Construct Graph.** The module responsible for constructing a binary graph adjacency matrix using the information about associations between database tables (data-items) manifested in the OLAP queries submitted by the users. More specifically, the data-item – data-item association between two database tables co-occurring in a join query is modeled using an edge between them in the constructed graph.
- **Calculate Edge Weights.** This module employs the use of query set characteristics to assign weights to edges constructed in the aforementioned step. Edge weights capture the strength of data-item – data-item associations, thereby appropriately accounting for the contribution of each edge towards minimizing the objective for CDR-Single.
- **Construct Hypergraph.** The module responsible for constructing a binary hypergraph incidence matrix using the information about a variety of associations between OSN users (data-items) and nodes manifested in the check-ins registered by these users. More specifically, the data-item – data-item association between OSN users co-occurring in a data-request pattern triggered by a user check-in is modeled using a hyperedge connecting data-items in the constructed hypergraph. In the same vein, the data-item – node association is modeled using a hyperedge connecting the data-item with the node location where the data-item was requested based on the user check-in.
- **Calculate Hyperedge Weights.** This module is responsible for assigning weights to each hyperedge of

[4]The weights determine the relative importance of these metrics towards the overall optimization objective, and are discussed in Sec. IV-B.

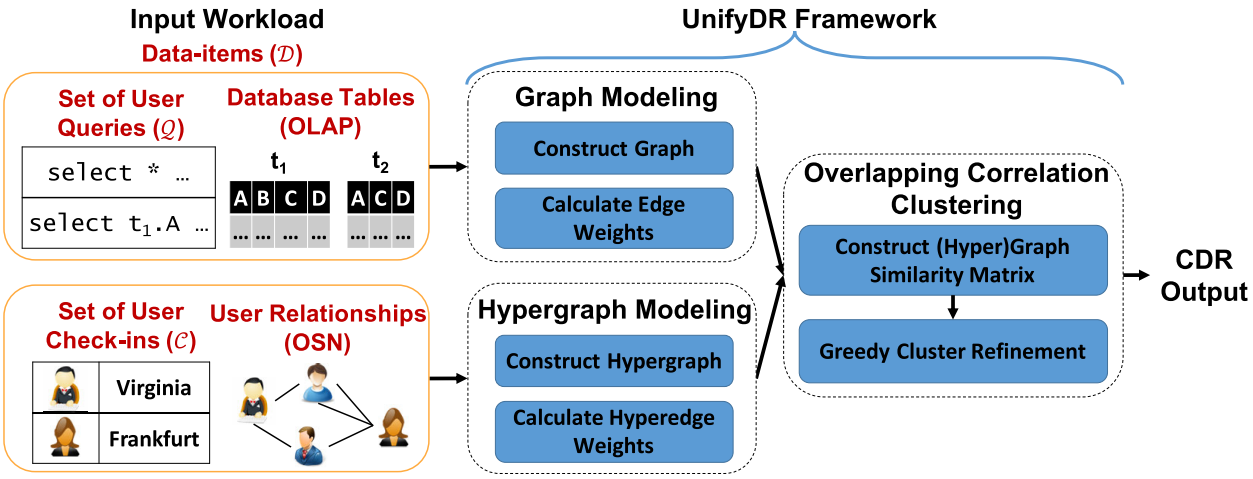

FIGURE 4. Overview of the proposed UnifyDR framework for combined data and replica placement.

the hypergraph constructed in the aforementioned step based on user check-in behavior and node characteristics. Hyperedge weights capture the strength of data-item – node and data-item – data-item associations, thereby enabling accurate estimations of the contribution of each hyperedge towards the multi-objective optimization for CDR-Multi.

- **Construct (Hyper)Graph Similarity Matrix.** This module uses the (hyper)graph representation and the (hyper)edge weights to compute the similarity between each pair of vertices in the (hyper)graph. It is required for performing analytical operations such as clustering on (hyper)graphs. Mathematical details of this step for both graphs and hypergraphs are provided in Sec. V.
- **Greedy Cluster Refinement.** This module facilitates partitioning the data-items into l nodes while allowing at most r replicas. Specifically, it iteratively refines the cluster assignment of each data-item given the cluster assignment of all the other data-items is fixed.

A. CDR-SINGLE: GRAPH MODELING

Data placement research has shown that graph partitioning is capable of accurately optimizing the objective of minimizing the number of nodes spanned by data-items partitioned across nodes [12], [17], [28]. Since the aim of CDR-Single is to minimize the number of nodes spanned during the execution of OLAP queries, graphs provide a powerful representation to model data-item – data-item associations in this case.

To this end, given a set of database tables $D(T)$ corresponding to data-items, and a set of queries $\mathcal{Q}$ representing the system workload, we construct a graph $G(V_G, E_G)$. There exists a vertex $v \in V_G$ for each data-item $d(t) \in \mathcal{D}$, thus, the vertex set V_G consists of $|V_G| = n$ vertices. Furthermore, there exists an edge between each pair of data-items (corresponding to database tables) that co-occur in a join query, thereby capturing data-item – data-item associations. Recall that for each query $Q_k \in \mathcal{Q}$, $\mathbf{QS}(Q_k)$ denotes the set of data-items (tables) that should co-exist for execution of the query. In other words, there should be an edge between each pair of vertices (corresponding to data-items) in $\mathbf{QS}(Q_k)$. With this, we formally define the edge set $E_G = \{(u, v)\} \mid u, v \in V_G \wedge \exists Q_k \in \mathcal{Q} : d(u), d(v) \in \mathbf{QS}(Q_k)$, totaling to $|E_G| = m$ edges.

The graph $G(V_G, E_G)$ is represented using a $n \times n$ dimensional binary adjacency matrix $\mathcal{A}$, which possesses n vertices and m edges. An entry $A_{i,j} = 1$ indicates that there is an edge between the i^{th} and j^{th} vertex in the graph vertex set, while $A_{i,j} = 0$ indicates otherwise.

While each edge $(u, v) \in E_G$ captures the association between two vertices u and v, not all associations are equally important. Instead, some associations are relatively more important. For example, two data-items $d(t_1)$ and $d(t_2)$ that co-occur in 10 join queries possess a stronger association when compared to another pair of data-items $d(t_1)$ and $d(t_3)$ that co-occur only twice. Thus, partitioning the edge between $d(t_1)$ and $d(t_2)$ should possess a higher cost when compared to that of the edge between $d(t_1)$ and $d(t_3)$. To capture this, we construct an edge weight matrix $\mathcal{W}_\mathcal{A}$ of dimensionality $n \times n$ that captures the relative importance of the edges. Each entry $\mathcal{W}_{\mathcal{A}i,j}$ captures the number of times the data-items corresponding to the i^{th} and j^{th} vertex co-occurred in a join query $Q_k \in \mathcal{Q}$.

In sum, the graph modeling step facilitates capturing the interaction between data-items in the form of a graph adjacency matrix $\mathcal{A}$, and the edge weight matrix $\mathcal{W}_\mathcal{A}$ representing the relative importance of the constructed edges.

B. CDR-MULTI: HYPERGRAPH MODELING

The suitability of hypergraphs to model the associations emanating from data-item – node and data-item – data-item interactions has been substantiated in a plethora of works [3], [4], [43] on geo-distributed data placement. As opposed to edges (in traditional graphs) that can only model pairwise relationships, hyperedges possess the capability of

modeling multi-way relationships by connecting several vertices together using a single hyperedge. Thus, in all respects, hypergraphs serve as a generalization over graphs, and being a more sophisticated construct, provide a solid representation for modeling data-item – data-node and data-item – data-item associations.

The set E_H of hyperedges constructed based on the information manifested in the check-ins registered by the OSN users consists of two types of hyperedges. (1) Data-request pattern hyperedges ($\mathcal{R}$): For each data-request triggered corresponding to a user check-in, a data-request pattern hyperedge captures data-item – data-item associations by connecting all of its constituent data-items via a single hyperedge; and (2) Data-item – node hyperedges ($\mathcal{R}_\mathcal{N}$): This hyperedge captures data-item – node associations by connecting each data-item contained in the data-request triggered in response to a user check-in to the node location of the registered check-in. The set of data-items $\mathcal{D}$ and nodes $\mathcal{N}$ constitute the hypergraph vertex set V_H, resulting in a total of $|V_H| = n' = n + l$ vertices. Similarly, the hypergraph edge set E_H is composed of the data-request pattern hyperedges $\mathcal{R}$ and the data-item – node hyperedges $\mathcal{R}_\mathcal{N}$, totaling to $|E_H| = m' = |\mathcal{R}| + nl$ hyperedges. These two sets are formally defined as follows.

$$\begin{aligned} V_H &= \mathcal{D} \cup \mathcal{N}, \\ E_H &= \mathcal{R} \cup \mathcal{R}_\mathcal{N}. \end{aligned} \tag{1}$$

We use a binary hypergraph incidence matrix Π with n' rows and m' columns to represent the constructed hypergraph $H(V_H, E_H)$. Moreover, each hyperedge is represented via a $n' \times 1$ binary column vector and the hypergraph contains a total of m' hyperedges. Mathematically,

$$\begin{aligned} \forall he_i \in E_H, he_i^T &= [he_{1,i}, he_{2,i}, \ldots, he_{n',i}], \\ \Pi &= [he_1, he_2, \ldots, he_{m'}]. \end{aligned} \tag{2}$$

An entry $he_{j,i} = 1$ indicates that the j^{th} vertex in the hypergraph vertex set is participating in the i^{th} hyperedge, while $he_{j,i} = 0$ indicates otherwise.

Moving further, we discuss the hyperedge weight assignment mechanisms, which lie in two broad categories corresponding to the aforementioned hyperedge types. These hyperedge weights guide the optimization by focusing on different objectives of the underlying optimization problem. For instance, to minimize the node span $\mathcal{S}(R(v))$, which is computed as the average number of node accesses for fetching the data-items requested in a data-request pattern $R(v)$, $W_\mathcal{R}$ enforces these co-occurring data-items in $R(v)$ to be placed in the same node by giving higher weight to data-request pattern hyperedges. In the same vein, to minimize the outgoing traffic cost $\Gamma(N_j)$, storage cost $\Delta(N_j)$, and inter node latency $\kappa(N_j, N_{j'})$, a higher preference is given towards placement of data-items at the nodes from where they were requested more often, by employing the use of data-item – node hyperedge weights $W^\Gamma_{\mathcal{R}_\mathcal{N}}$, $W^\Delta_{\mathcal{R}_\mathcal{N}}$, and $W^\kappa_{\mathcal{R}_\mathcal{N}}$, respectively. Eventually, we perform a weighted sum of the four weighting mechanisms presented above to obtain the final hyperedge weight matrix. Mathematically,

$$\mathcal{W}_\Pi = \mathbb{W} \cdot (W_\mathcal{R}, W^\kappa_{\mathcal{R}_\mathcal{N}}, W^\Delta_{\mathcal{R}_\mathcal{N}}, W^\Gamma_{\mathcal{R}_\mathcal{N}}). \tag{3}$$

where, the vector $\mathbb{W}$ controls the relative importance of the aforementioned hyperedge weight assignment strategies[5] to obtain the final diagonal hyperedge weight matrix $\mathcal{W}_\Pi$ of size $m' \times m'$.

To summarize, the hypergraph modeling step produces a hypergraph incidence matrix Π and a hyperedge weight matrix $\mathcal{W}_\Pi$, where the former (Π) models the higher-order interaction between data-items and nodes, while the latter ($\mathcal{W}_\Pi$) controls the relative importance of different hyperedges.

C. OVERLAPPING CORRELATION CLUSTERING

Overlapping clustering has been shown to possess applications in a wide-variety of research areas: community detection [29], bioinformatics [6], and information retrieval [6]. The key feature of overlapping clustering is that it allows each data point to be assigned to more than one cluster, which is a natural requirement for multiple real-world applications. For example, it is highly likely for a user to be a part of more than one community. Having motivated the importance of overlapping clustering, we next provide a brief summary of the steps required to perform overlapping correlation clustering on (hyper)graphs.

The first step requires construction of a (hyper)graph similarity matrix, where each entry corresponds to a similarity score between two vertices. Moving ahead, each vertex is randomly assigned to at most l different clusters. The next step is to greedily refine the cluster assignments of each vertex (given the assignments of all other vertices is fixed), with the objective that the similarity between any two vertices agree as much as possible with the similarity computed based on their cluster assignments. This two step process: (1) (Hyper)Graph similarity computation, followed by (2) Greedy cluster refinement, constitutes the proposed combined data and replica placement algorithm.

V. COMBINED DATA AND REPLICA PLACEMENT ALGORITHM

In this section, we first describe the overlapping correlation clustering algorithm in a detailed and formal manner. Next, we describe *OverlapG* and *OverlapH*, the combined data and replica placement algorithms proposed in this work.

Given the set of user queries $\mathcal{Q}$ and user check-ins $\mathcal{C}$ representing the system workload for the OLAP (CDR-Single) and the OSN (CDR-Multi) use-case, respectively, and the set of data-items $\mathcal{D}$ as input, the first step is to construct a graph (line 2 in OverlapG) or a hypergraph (line 8 in OverlapH). This is followed by the construction of normalized graph (lines 3–4 in OverlapG) or hypergraph (lines 9–10 in

[5]Please see [3], [5], [43] for further details pertaining to the construction of hypergraph and hyperedge weight assignment.

Algorithm 1 CDR Placement Algorithm

Input: $\mathcal{D}, \mathcal{R}, \mathcal{N}, r, l, \mathcal{Q}, \mathcal{C}$

Output: Partitioning of the set of data-items $\mathcal{P}(\mathcal{D})$ into l nodes allowing r replicas

1: **procedure** OverlapG($\mathcal{D}, \mathcal{R}, \mathcal{N}, r, l, \mathcal{Q}$) ▷ OLAP Use-case
2: $(\mathcal{A}, \mathcal{W}_{\mathcal{A}}) \leftarrow$ ConstructGraph($\mathcal{D}, \mathcal{R}, \mathcal{N}, \mathcal{Q}$)
3: $D_{v\mathcal{A}} \leftarrow diag(\sum \mathcal{A})$
4: Compute normalized graph G^{sim} as described in Eq. (9)
5: $\mathcal{P}(\mathcal{D}) \leftarrow$ OverlappingCorrelationClustering (V, N_G, l, r)
6: **end procedure**
7: **procedure** OverlapH($\mathcal{D}, \mathcal{R}, \mathcal{N}, r, l, \mathcal{C}$) ▷ OSN Use-case
8: $(\Pi, \mathcal{W}_{\Pi}) \leftarrow$ ConstructHypergraph($\mathcal{D}, \mathcal{R}, \mathcal{N}, \mathcal{C}$)
9: $D_{v\Pi} \leftarrow diag(\sum \Pi)$; $D_{he\Pi} \leftarrow diag(\sum \Pi^T)$
10: Compute normalized hypergraph H^{sim} as described in Eq. (12)
11: $\mathcal{P}(\mathcal{D}) \leftarrow$ OverlappingCorrelationClustering (V_H, N_H, l, r)
12: **end procedure**
13: **return** $\mathcal{P}(\mathcal{D})$

OverlapH) similarity matrix depending upon the use-case. The last step employs the use of the proposed overlapping clustering algorithm (line 5 in OverlapG and line 11 in OverlapH) to assign each data-item $d(t)$ or $d(v) \in \mathcal{D}$ to $r < l$ nodes, thereby obtaining a partitioning of $\mathcal{D}$ while allowing for at most r replicas per data-item.

A. PRELIMINARIES

As a first step, we provide a description of correlation clustering. Given a normalized similarity matrix M^{sim} representing the pair-wise similarity between data-items, and a set $\mathcal{N}$ of l labels representing nodes as input, the task of correlation clustering is to find a mapping $\mathcal{F} : V \rightarrow \mathcal{N}$ for partitioning the set of data-items into l nodes, that minimizes the following loss function:

$$\mathbb{L}_{Correlate}(V, \mathcal{F}) = \sum_{\substack{(u,v)\in V\times V \\ \mathcal{F}(u)=\mathcal{F}(v)}} (1 - M^{sim}(u, v)) + \sum_{\substack{(u,v)\in V\times V \\ \mathcal{F}(u)\neq\mathcal{F}(v)}} M^{sim}(u, v). \quad (4)$$

Moving ahead, overlapping correlation clustering was introduced to relax the requirement of correlation clustering to assign each data-item to exactly one partition. This design choice is well-motivated. For instance, in social networks a user might be a part of multiple communities. In the context of data placement, replication of data-items is often required for ensuring fault-tolerance and obtaining a lower overall communication cost.

Having said that, we leverage this feature of overlapping clustering to obtain a partitioning of $\mathcal{D}$ into l nodes by assigning each data-item to multiple nodes, thereby allowing for replication. In overlapping clustering, this is achieved by mapping each data-item to a label set as opposed to a single label, where each label corresponds to a node. Given the label set definition as the set of all subsets of nodes $\mathcal{N}$ except the empty set: $\mathcal{N}^+ = 2^{\mathcal{N}} \setminus \{\emptyset\}$, and a similarity function over the data-item label sets $\mathbb{S}(\cdot)$, the underlying optimization objective reduces to identifying a mapping $\mathcal{F} : V \rightarrow \mathcal{N}^+$ under which the similarity between any pair of data-items $\forall u, v \in V$, $M^{sim}(u, v)$ agrees as much as possible with the similarity between their corresponding label sets $\mathbb{S}(\mathcal{F}(u), \mathcal{F}(v))$.

Similar to the loss function for correlation clustering $\mathbb{L}_{Correlate}$, the loss function for overlapping correlation clustering is defined as:

$$\begin{aligned}\mathbb{L}_{Overlap}(V, \mathcal{F}) &= \sum_{(u,v)\in V\times V} |\mathbb{S}(\mathcal{F}(u), \mathcal{F}(v)) - M^{sim}(u, v)|, \\ &= \sum_{u\in V} \sum_{v\in V\setminus\{u\}} |\mathbb{S}(\mathcal{F}(u), \mathcal{F}(v)) - M^{sim}(u, v)|. \end{aligned} \quad (5)$$

where $\mathbb{S}(\cdot)$ is defined as the set-intersection indicator function:

$$\mathbb{S}(X, Y) = \begin{cases} 1, & \text{if } X \cap Y \neq \emptyset. \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Formally, the goal of overlapping correlation clustering is to find a mapping $\mathcal{F}^*$ in order to minimize $\mathbb{L}_{Overlap}(V, \mathcal{F})$, which is mathematically denoted as:

$$\mathcal{F}^* = \underset{\mathcal{F}}{\operatorname{argmin}}\, \mathbb{L}_{Overlap}(V, \mathcal{F}). \quad (7)$$

Note that for the OLAP use-case the underlying representation is a graph, hence the set of vertices is represented using V_G while the normalized similarity matrix M^{sim} is represented using G^{sim}. Similarly, for the OSN use-case the set of vertices is represented using V_H while the normalized hypergraph similarity matrix is denoted as H^{sim}.

As stated in Sec. IV, overlapping correlation clustering requires a similarity matrix denoting similarities between each pair of vertices in the (hyper)graph as input. Thereby, we provide a formal description of the graph and hypergraph similarity matrix construction in the following sections.

B. CDR-SINGLE: NORMALIZED GRAPH SIMILARITY MATRIX

To construct the normalized graph similarity matrix G^{sim}, we first compute the vertex degree matrix ($D_{v\mathcal{A}}$) from the graph adjacency matrix $\mathcal{A}$. $D_{v\mathcal{A}}$ is a diagonal matrix of dimensionality $n \times n$, which captures the number of adjacent vertices for each vertex in the graph. Mathematically,

$$D_{v\mathcal{A}} = diag(\sum \mathcal{A}). \quad (8)$$

Algorithm 2 Overlapping Clustering Algorithm

Input: V, M^{sim}, l, r

Output: Partitioning of the (hyper)graph vertex set $\mathcal{P}(V)$ into l clusters allowing r replicas

1: Randomly initialize the label sets of size r for each data-item $u \in V$
2: **while** $\mathbb{L}_{Overlap}(V, \mathcal{F})$ decreases **do**
3: **for each** $u \in V$ **do**
4: find the label set F that minimizes $\mathbb{L}^{u}_{Overlap}(F|\mathcal{F})$
5: Update $\mathcal{F}(u) \leftarrow F$
6: **end for**
7: **end while**
8: **return** $\mathcal{P}(V)$ defined by $\mathcal{F}$

where, $\sum X$ represents the row-wise sum of the input matrix X.

Note that the methods for normalizing a similarity matrix was described by [32], [40] for constructing the graph laplacian. To this end, we mathematically define the normalized graph similarity matrix G^{sim} as follows.

$$G^{sim} = \left(D_{v\mathcal{A}}^{-1/2} \cdot \mathcal{A} \cdot \mathcal{W}_{\mathcal{A}} \cdot D_{v\mathcal{A}}^{-1/2}\right). \tag{9}$$

where, $\mathcal{W}_{\mathcal{A}}$ is a $n \times n$ edge weight matrix. Thus, G^{sim} becomes a $n \times n$ matrix.

C. CDR-MULTI: NORMALIZED HYPERGRAPH SIMILARITY MATRIX

A main requirement for constructing the normalized hypergraph similarity matrix H^{sim} is the computation of two diagonal degree matrices from the hypergraph incidence matrix Π. (1) The vertex degree matrix $D_{v\Pi}$ of size $n' \times n'$, which measures the number of hyperedges that are connected to each vertex, and (2) the hyperedge degree matrix $D_{he\Pi}$ of size $m' \times m'$, which captures the number of vertices that are connected together by each hyperedge. The two degree matrices are mathematically defined as follows.

$$D_{v\Pi} = diag(\sum \Pi). \tag{10}$$

$$D_{he\Pi} = diag(\sum \Pi^{T}). \tag{11}$$

where, $\sum X$ represents the row-wise sum of the input matrix X and X^T represents the transpose of the matrix X.

Similar to the normalization procedure for a similarity matrix [32], [40], the normalization procedure for hypergraphs was defined in [50]. To this end, we mathematically define the normalized hypergraph similarity matrix H^{sim} as follows.

$$H^{sim} = \left(D_{v\Pi}^{-1/2} \cdot \Pi \cdot \mathcal{W}_{\Pi} \cdot D_{he\Pi}^{-1} \cdot \Pi^{T} \cdot D_{v\Pi}^{-1/2}\right). \tag{12}$$

where, $D_{v\Pi}$ and $D_{he\Pi}$ are diagonal matrices of size $n' \times n'$ and $m' \times m'$ storing vertex and hyperedge degrees, respectively. Moreover, $\mathcal{W}_{\Pi}$ is a $m' \times m'$ diagonal matrix representing hyperedge weights. With this, the dimensionality of the normalized hypergraph similarity matrix H^{sim} becomes $n' \times n'$.

Having described the (hyper)graph similarity matrix construction, we next discuss the optimization approach for performing overlapping clustering.

D. GREEDY CLUSTER REFINEMENT

Overlapping correlation clustering cannot be solved optimally in polynomial time as it is a NP-Hard problem [6]. Thus, we employ the use of an iterative greedy algorithm focused on improving the label set quality of one vertex at a time. More specifically, keeping the label sets of all the other vertices in the (hyper)graph as fixed, the greedy algorithm applies a local optimization (on one vertex) to improve the cost of the overall solution until convergence. The steps for performing overlapping correlation clustering are portrayed in Algorithm 2.

The first step requires initializing each vertex $u \in V$ with a set of cardinality r consisting of randomly assigned labels (line 1). This initialization allows each data-item to be simultaneously assigned to r different nodes. Next, the aforementioned greedy local optimization approach is applied to iteratively refine the label sets of each vertex (lines 2–7). More specifically, the label set of each node $u \in V$ is iteratively improved while keeping the label sets of all the other nodes fixed, until the overall loss $\mathbb{L}_{Overlap}(V, \mathcal{F})$ converges. Note that Eq. (5) is reformulated to clearly depict the loss with respect to each node u, and is stated as follows.

$$\mathbb{L}(V, \mathcal{F}) = \sum_{u \in V} \mathbb{L}^{u}_{Overlap}(\mathcal{F}(v) \mid \mathcal{F}). \tag{13}$$

where,

$$\mathbb{L}^{u}_{Overlap}(\mathcal{F}(v) \mid \mathcal{F}) = \sum_{v \in V \setminus \{u\}} |\mathbb{S}(\mathcal{F}(u), \mathcal{F}(v)) - M^{sim}(u, v)|. \tag{14}$$

VI. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed UnifyDR framework in solving the CDR-Single and CDR-Multi problems through experiments on data simulated using a decision support benchmark, and data extracted from a large scale location-based OSN respectively. We perform experiments with algorithms implemented in C++ on an Intel(R) Xeon(R) E5-2680 v3 24-core machine with 2.5 GHz CPU and 256 GB RAM running Linux Ubuntu 18.04. Results corresponding to all the methods (barring Nearest) are averaged over 10 runs.

A. CDR-SINGLE

1) DATASET

In accordance with the state-of-the-art for data placement research in scientific workflows [17], we incorporate the use of data simulated based on the TPC[6] decision support

[6]The transaction processing performance council (TPC) is a non-profit organization with the focus of defining benchmarks for transaction processing in databases.

(TPC-DS) benchmark [11]. The TPC-DS benchmark provides an appropriate medium to study the OLAP use-case as it facilitates modeling of various performance aspects of a decision support system, such as query execution and data maintenance. The benchmark contains a total of 24 database tables (7 fact and 17 dimension tables) corresponding to data-items. Additionally, it contains 99 queries that represent the system workload.

2) SETUP

A real-world distributed execution environment is simulated by partitioning the 24 data-items in l nodes. The number of nodes l is varied from 2 to 8. Since usually the fact tables are significantly larger than the dimension tables, the size of the data-items corresponding to the fact tables is chosen from the range [50, 100] units, while those corresponding to the dimension tables is chosen from the range [1, 10]. Furthermore, let the total size of all the data-items be $S = \sum_{t \in T} size(d(t))$ and the size of the largest data-item be S_{max}, then the storage capacity of each node $N_j \in \mathcal{N}$ is set as $max(S/l, S_{max})$. This is because the storage capacity of a node is dependent upon the total size of all the data-items as well as the size of the largest data-item.

In summary, the CDR placement task for the OLAP use-case reduces to partitioning 24 data-items corresponding to the database tables into {2, 4, 8} nodes respectively based on the data-request patterns triggered from 99 user queries.

Baselines: We considered the following two baselines.

- **Random:** obtains a placement by randomly partitioning the set of data-items $\mathcal{D}$ into $|\mathcal{N}|$ nodes.
- **Bipartite graph partitioning (Metis):** is the placement algorithm proposed in [17], which constructs a bipartite graph of the set of queries $\mathcal{Q}$ and the set of tables $D(T)$, and employs Metis [25] to perform graph partitioning. The authors tackle replica placement using a heuristic termed H2.

As discussed in Sec. II, Metis (with the replication heuristic H2) [17] serves as the representative state-of-the-art method for data placement and replication of data-intensive scientific workflows.

Parameters: The parameters for Metis are set based on the recommendations by the authors of [17], where k-way partitioning with 1000 cuts and 1000 iterations is used. Lastly, we fix the replication factor r to 3 based on best practices prescribed in the field of data storage management [20].

Evaluation Metrics: The metrics used to evaluate the effectiveness of OverlapG in solving the CDR-Single problem are stated as follows:

- **Efficiency:** The execution time, which is measured as the time required to obtain CDR placement, is used to evaluated the efficiency of the methods benchmarked in this study.
- **Efficacy:** is measured using the node span $\mathcal{S}(\cdot)$ of data-request patterns, which is defined as the average number of node accesses required to fetch the data-items requested in a query Q_k corresponding to a request pattern $R(Q_k)$. The average of the node spans of each data-request pattern $\exists Q_k \in \mathcal{Q} : R(Q_k) \in \mathcal{R}$ represents the span of the entire system workload.

Note that we normalize the node spans obtained from different techniques in the scale of [0, 1] as it provides an intuitive way to analyze their relative performance. Furthermore, since the optimization problem underneath CDR-Single is concerned with the minimization of node span, smaller values imply better performance.

3) RESULTS

Figs. 5a– 5d present the results for the CDR-Single problem on the considered evaluation metrics. We compare OverlapG with the considered baselines on two settings: (1) no replication ($r = 0$), and (2) with replication ($r = 3$). This analysis is performed to showcase the backward compatibility of overlapping clustering. In other words, it indicates the ability of overlapping clustering to assign each data-item to one and only one node. Fig. 5a shows that OverlapG produces a node span similar to Metis, which is the current state-of-the-art. Furthermore, both OverlapG and Metis are significantly better when compared to the Random baseline, thereby portraying their ability to effectively capture data-item – data-item associations. Having said that, we also compare the execution time of OverlapG with Metis under this setting. Fig. 5c shows that both techniques require a similar amount of time, 2.3 – 4.6 seconds for Metis and 2.6 – 4.9 seconds for OverlapG, to perform data placement. Note that the execution time of OverlapG is slightly on the higher side owing to the relatively higher complexity of the optimization.

Having analyzed the ability of OverlapG to perform data placement (without replication $r = 0$), we next evaluate its ability to effectively solve the CDR-Single problem. As discussed previously, for this analysis we consider the setting where each data-item can possess at most 3 replicas ($r = 3$). It is evident from Fig. 5b that OverlapG significantly outperforms the Random baseline, while also achieving a reduction of around 30% in node span when compared to Metis+H2. This reduction is attributed to the ability of OverlapG to jointly optimize for data and replica placement in a unified manner. On the contrary, the two-step approach of performing data placement using Metis followed by replica placement using a heuristic (H2) falls short of finding a high quality solution to the CDR-Single problem. Furthermore, as shown in Fig. 5d it is interesting to note that OverlapG achieves a speed-up of around 20% over Metis+H2 as well. While OverlapG possessed a slightly higher execution time when compared to Metis for $r = 0$, it is faster when compared to Metis+H2 for $r = 3$. The main reason behind this is the requirement of executing two algorithms (Metis and H2) on the entire workload for the latter.

In summary, our experimental evaluation portrays the *effectiveness* of OverlapG in solving the CDR-Single problem. Additionally, its ability to perform data and replica placement in a single step allows for a better and unified system design.

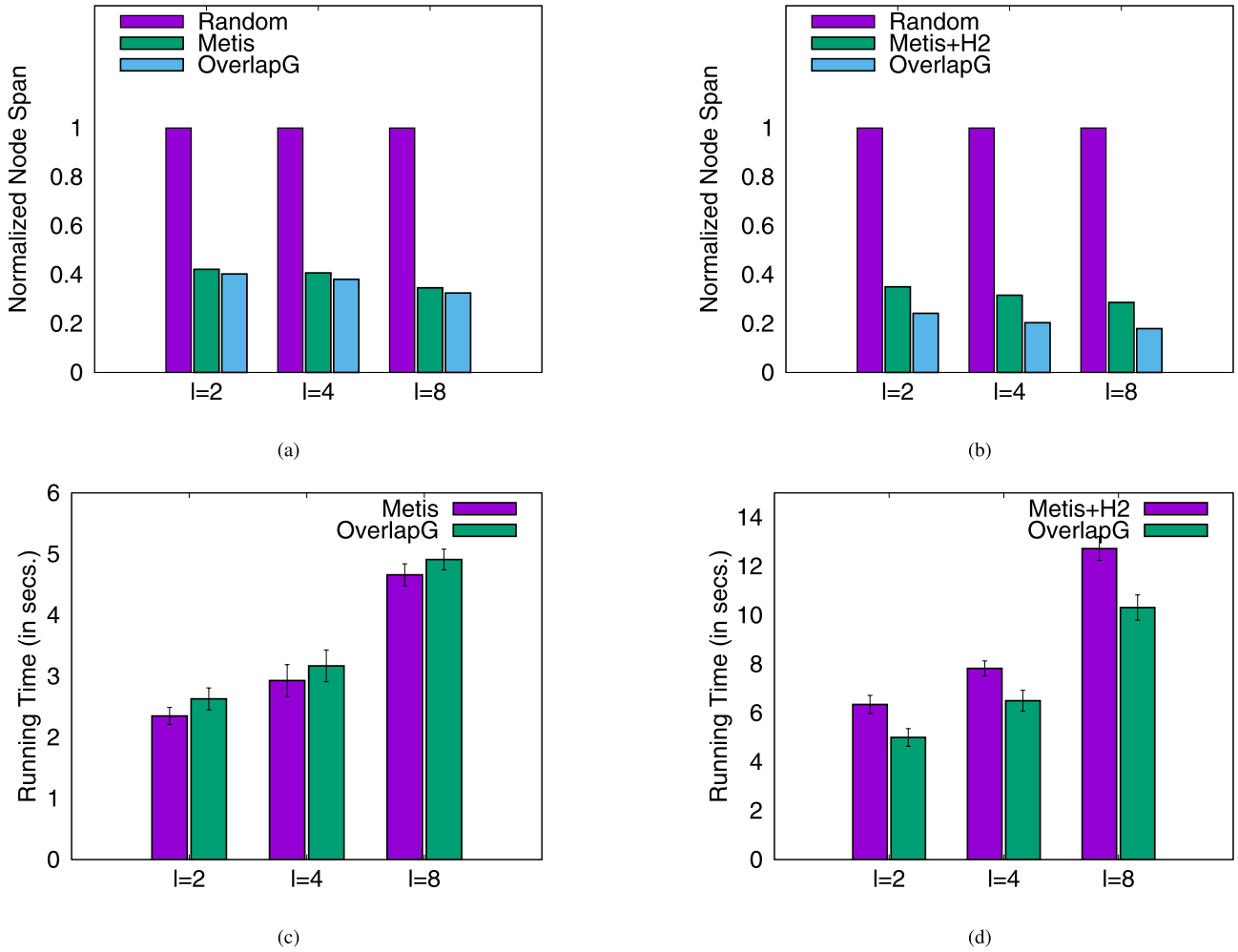

FIGURE 5. Analyzing the performance of OverlapG on the considered evaluation metrics. OverlapG achieves (a) a similar node span $\mathcal{S}(\cdot)$ as Metis when $r = 0$; while (b) it results in a reduction of $\approx 31\%$ when compared to Metis+H2 with $r = 3$. The execution time of OverlapG is (c) similar to Metis for $r = 0$, while it is (d) ≈ 1.3 times faster than Metis+H2 for $r = 3$.

B. CDR-MULTI

1) DATASET

We use the publicly available *Gowalla*[7] social network dataset, as it is a popular choice in the data placement literature on geo-distributed cloud services [3], [43]. The dataset consists of 196591 social network users (represented via vertices in the social network), and 950327 friend relationships (represented via edges). Additionally, the dataset provides information pertaining to user behavior. It contains 6442890 check-ins registered by social network users between February, 2009 and October, 2010, triggering 102314 unique data-request patterns in total.

2) SETUP

The AWS global infrastructure [2] is used as a basis for simulating a geographically distributed cloud execution environment. Following convention in the literature [3], [41], the $l = 9$ oldest AWS data center (node) regions[8]—California, Frankfurt, Ireland, Oregon, Sao Paulo, Singapore, Sydney, Tokyo, and Virginia—are used to setup our experiments. The traffic and storage costs for each node are set as advertised by Amazon. Furthermore, we use the Linux *ping* command [38] to obtain the packet transfer latency between the chosen node regions, which provides a good estimate of their inter node latency. The aforementioned steps enable us to closely mimic the real AWS execution environment. Table 1 presents the node characteristics.

Based on our analyses, we identified the existence of disparity in the user check-in behavior. More specifically, while there exist nodes that receive a huge number of check-ins (e.g., Frankfurt and Virginia), there are others where only a few check-ins are registered (e.g., Sydney and Sao Paulo). Having said that, both the number and the size (measured as the size of the triggered data-request pattern) of the

[7]http://snap.stanford.edu/data/loc-gowalla.html

[8]These regions are listed in alphabetical order.

TABLE 1. (a) Traffic and Storage Costs, and (b) Inter Node Latency based on Geo-distributed Amazon Clouds.

Region	Storage ($/GB-month)	Outgoing Traffic ($/GB)
California	0.026	0.02
Frankfurt	0.025	0.02
Ireland	0.023	0.02
Oregon	0.023	0.02
Sao Paulo	0.041	0.16
Singapore	0.025	0.02
Sydney	0.025	0.14
Tokyo	0.025	0.09
Virginia	0.023	0.02

(a) Costs (in $)

Region	California	Frankfurt	Ireland	Oregon	Sao-Paulo	Singapore	Sydney	Tokyo	Virginia
California	5.842	166.609	153.202	19.464	192.670	174.010	157.463	102.504	71.632
Frankfurt	166.590	4.425	19.533	159.542	194.905	325.934	323.483	236.537	88.624
Ireland	153.220	19.560	5.005	136.976	191.292	239.023	309.562	212.388	80.524
Oregon	19.204	159.523	136.979	5.551	182.716	161.367	162.175	89.095	88.683
Sao-Paulo	192.700	194.900	191.559	181.665	6.076	327.924	322.523	256.665	119.542
Singapore	173.946	325.918	238.130	161.423	328.080	5.870	175.328	73.807	216.680
Sydney	157.843	323.152	309.562	161.932	322.494	175.355	4.889	103.900	229.748
Tokyo	102.523	236.558	212.388	89.157	256.763	73.785	103.907	6.846	145.261
Virginia	72.738	88.657	80.546	86.981	119.531	216.719	229.972	145.255	3.523

(b) Latency (in ms)

check-ins registered in a region dictate the storage capacity required at each node. The storage capacity of each node $\forall N_j \in \mathcal{N}$ is therefore computed as $S_j = \sum |R(v)| : \exists C_k = (R(v), L_j), L_j = N_j.loc \in \mathcal{L}$.

In summary, the CDR placement task for the OSN use-case reduces to distributing 196591 data-items corresponding to OSN users into 9 nodes based on 102314 data-request patterns triggered from user check-ins.

Baselines: We consider the following four baselines.

- **Random:** obtains a placement by randomly partitioning the set of data-items $\mathcal{D}$ into $|\mathcal{N}|$ nodes.
- **Nearest:** produces a placement by assigning each data-item to the node with the highest number of requests for that data-item.
- **Hypergraph Partitioning (Hyper):** partitions the hypergraph induced on data-items and nodes using algorithms from the PaToH toolkit [7] to produce the data placement output. Hyper was proposed in [41], [43].
- **Spectral Clustering (Spectral):** obtains a placement using spectral clustering on hypergraphs and achieves superior efficiency by leveraging randomized eigendecomposition methods. Spectral was proposed in [5].

Recall that as described in Sec. II, the representative state-of-the-art for data placement and replication of data-intensive services into geographically distributed clouds is comprised of the techniques Spectral [5] and Hyper [43].

Parameters: The weight vector $\mathbb{W}$ (Eq. (3)) is one of the most important parameters for tuning the optimization underlying Spectral, Hyper, and OverlapH algorithms. It enables preferred optimization of certain chosen evaluation metrics over others by controlling the relative importance of different hyperedge weights described in Sec. IV-B. For our experiments, we have employed the use of 4 different settings of the parameter $\mathbb{W}$, namely $\mathbb{W}_1 : \{100, 1, 1, 1\}$, $\mathbb{W}_2 : \{1, 100, 1, 1\}$, $\mathbb{W}_3 : \{1, 1, 100, 1\}$, and $\mathbb{W}_4 : \{1, 1, 1, 100\}$ for minimizing the node span $\mathcal{S}(\cdot)$, inter node traffic $\Gamma(\cdot)$, inter node latency $\kappa(\cdot)$, and storage cost $\Delta(\cdot)$, respectively. Note that the value 100 used to obtain different weight-vector settings $\mathbb{W}_1$–$\mathbb{W}_4$ is only indicative of the higher relative importance provided to the metric under consideration. Having said that, any value $\gg 1$ should facilitate reproducibility of the main findings in the presented results, as the observed trends do not depend on the chosen value 100. Unless stated otherwise, spectral clustering is performed by using the 100 smallest eigenvectors of the hypergraph laplacian. Similar to the parameter setting for CDR-Single (Sec. VI-A2), we fix the replication factor r to 3.

Evaluation Metrics: The metrics used to evaluate the effectiveness of OverlapH in solving the CDR-Multi problem are stated as follows.

- **Efficiency:** Similar to the CDR-Single case (Sec. VI-A2), efficiency of the methods is measured using their execution time.
- **Efficacy:** is measured using multiple metrics,[9] namely the node span $\mathcal{S}(\cdot)$ (Span), outgoing traffic cost $\Gamma(\cdot)$ (Traffic), inter node latency $\kappa(\cdot)$ (Latency), storage cost $\Delta(\cdot)$ (Storage), and a weighted sum (Obj) of the four aforementioned metrics using the weights prescribed by $\mathbb{W}$.

Similar to the CDR-Single case (Sec. VI-A2), we normalize the results obtained from different techniques corresponding to each evaluation metric in the scale of [0, 1] as it provides an intuitive way to analyze their relative performance. Furthermore, normalization ensures equal and fair contribution of each evaluation metric towards *Obj* as all values lie in the common range [0, 1]. Lastly, it is important to note that since the optimization problem underneath CDR-Multi is concerned with the minimization of the aforementioned evaluation metrics, smaller values imply better performance.

3) RESULTS: EFFICACY

It is evident from the results portrayed in Figs. 6a– 6d that OverlapH is the best performing technique in terms of the weighted sum of the considered metrics, i.e., Obj, which is observed across each of the four different weight vector settings $\mathbb{W}_1$–$\mathbb{W}_4$ considered in the evaluation. Specifically, OverlapH substantially outperforms Random and Nearest, and is around 30–40% and 20–30% better when compared to Hyper and Spectral, respectively.

Redirecting our focus to other evaluation metrics, it can be noticed that Nearest outperforms Hyper, Spectral, and OverlapH in some cases, however, the latter are still significantly better than the Random method. For instance consider Fig. 6a, it can be observed that Nearest is better on the inter node traffic and latency metrics. This is because according to the

[9]For details on these metrics, please see Sec. 5.2 of [4].

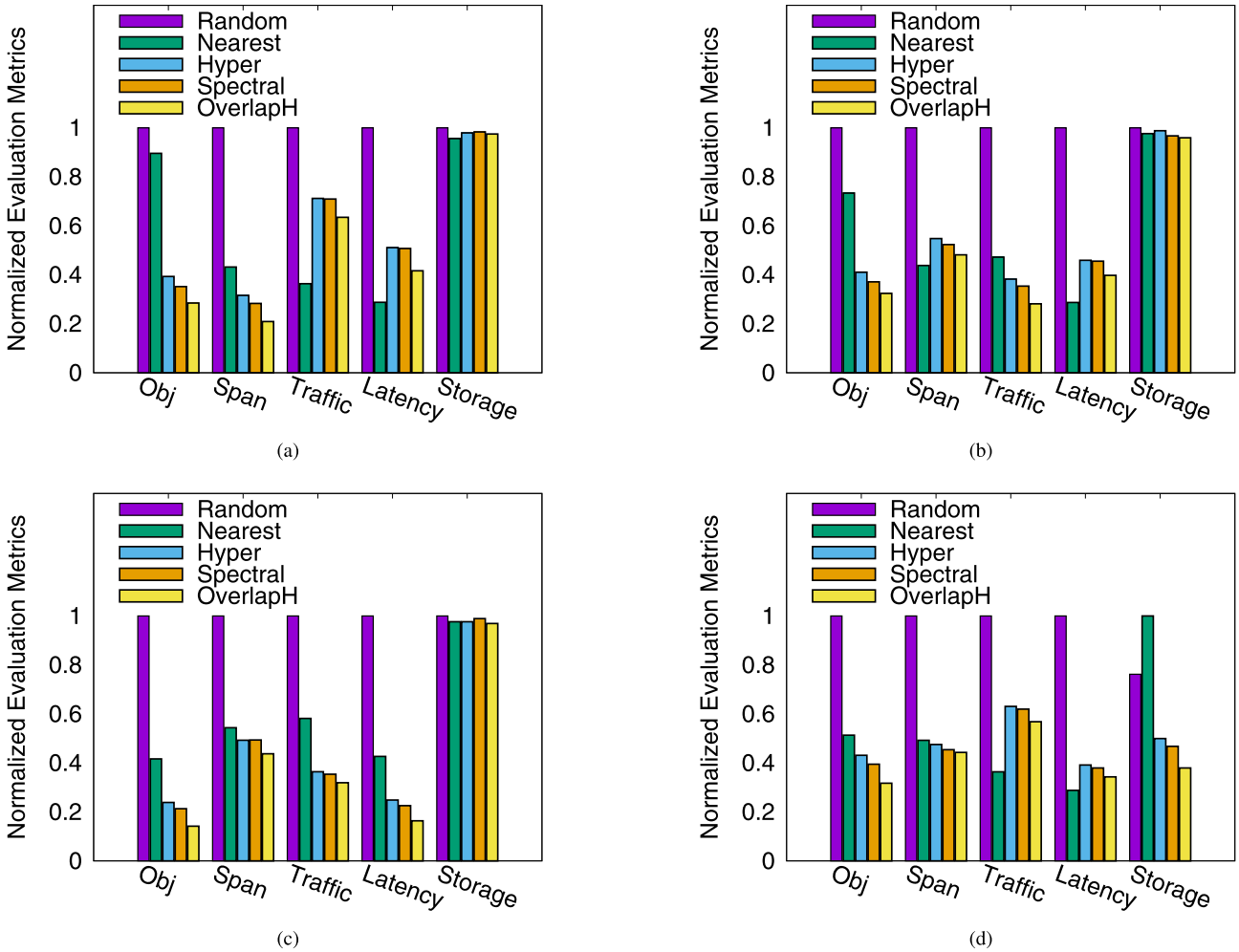

FIGURE 6. Analyzing the effect of different weight-vector settings on the evaluation metrics. OverlapH results in reducing the (a) node span $\mathcal{S}(\cdot)$ by $\approx$ 35% when compared to Spectral with $\mathbb{W}_1 = \{100, 1, 1, 1\}$; (b) inter node traffic $\Gamma(\cdot)$ by $\approx$ 26% when compared to Spectral with $\mathbb{W}_2 = \{1, 100, 1, 1\}$; (c) inter node latency $\kappa(\cdot)$ by $\approx$ 38% when compared to Spectral with $\mathbb{W}_3 = \{1, 1, 100, 1\}$; (d) storage cost $\mathcal{S}(\cdot)$ by $\approx$ 24% when compared to Spectral with $\mathbb{W}_4 = \{1, 1, 1, 100\}$.

weight vector setting $\mathbb{W}_1$, minimizing the node span holds the highest priority while traffic and latency metrics have lower weights in the optimization objective. A similar behavior is observed for the other three weight vector settings: $\mathbb{W}_2$, $\mathbb{W}_3$, and $\mathbb{W}_4$ as well (Figs. 6b– 6d). To understand this observed behavior better, let us analyze the results presented in Fig. 6d. It is not hard to infer that storage cost might be inversely related to other parameters such as inter node latency and traffic. Therefore, preferentially optimizing to achieve lower storage costs ($\mathbb{W}_4$) thereby also obtaining better performance on Obj, might lead a technique to suffer on other metrics, i.e., a lower storage cost might lead to higher latencies or traffic cost. Despite this behavior, most importantly OverlapH significantly outperforms all the considered baselines on the corresponding evaluation metric that the weight-vector setting is tuned to optimize. More fundamentally, in addition to being better on Obj, OverlapH outperforms the other methods in minimizing the node span $\mathcal{S}(\cdot)$, inter node traffic cost $\Gamma(\cdot)$, inter node latency $\kappa(\cdot)$, and storage cost $\Delta(\cdot)$, when a higher preference is given to these metrics under the weight-vector settings $\mathbb{W}_1$, $\mathbb{W}_2$, $\mathbb{W}_3$, and $\mathbb{W}_4$ respectively.

Moving ahead, we analyze the reason behind the sub-optimal performance of the Nearest method. The main limitation is that Nearest is inclined to assign each data-item to a node that receives the highest number of access requests for that data-item, which consequently results in minimizing (on an average) the geographical distance between the data-item and the source location of the data-request. Note that this optimization strategy is oblivious to the fact that the storage or traffic costs might not be correlated with the distance, thereby leading to sub-optimal performance in real-world settings that require multi-objective optimization. We also refer the reader to Table 2, which presents a quantitative summary of the performance of all the considered baselines indicating how worse each baseline is relative to OverlapH.

Based on the above analysis, it is clear that Hyper, Spectral, and OverlapH possess the capability to adapt the optimization based on the input weight vector setting. This is

TABLE 2. Quantifying the performance of the considered baselines relative to OverlapH on the evaluation metrics.

Algorithm	Degradation in performance of Baselines relative to OverlapH				
	Span	Traffic	Latency	Storage	Obj
Random	377.78%	255.24%	510.87%	100.87%	274.68%
Nearest	106.29%	67.70%	160.50%	163.64%	139.69%
Hyper	50.91%	35.78%	51.85%	31.67%	37.98%
Spectral	35.26%	25.65%	37.69%	23.28%	24.66%

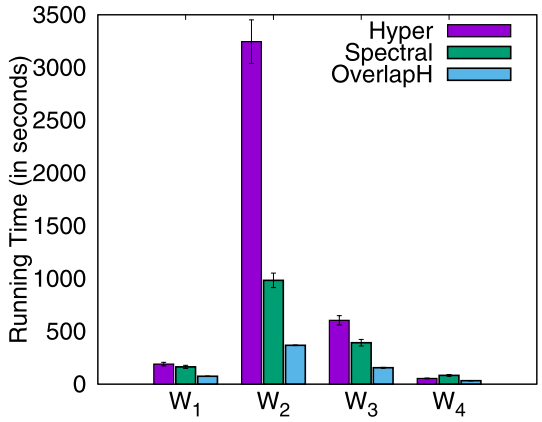

FIGURE 7. Comparing the execution time of OverlapH with spectral clustering [5] and hypergraph partitioning [43] algorithms.

because of their higher-order modeling capabilities courtesy hypergraphs, which renders them better suited for performing multi-objective optimizations. Further, since OverlapH models data placement and replication as a joint optimization problem (CDR-Multi), it achieves better performance on the evaluation metrics when compared to both Hyper and Spectral that solve each problem independently.

4) RESULTS: EFFICIENCY AND SCALABILITY

In this section, we analyze the execution time performance of Hyper, Spectral, and OverlapH—the three techniques that stand out in terms of performance on efficacy related evaluation metrics (Sec. VI-B3)—on the Gowalla dataset. It is evident from Fig. 7 that OverlapH substantially outperforms Hyper and Spectral in terms of execution time efficiency. Specifically, OverlapH achieves an average speed-up of $\approx$ 4–5 and $\approx$ 2–3 times over and above Hyper and Spectral, respectively. The capability of scaling to large datasets is a desired property in any CDR placement algorithm. OverlapH, with its ability to gracefully scale to large scale social networks, stands strong on this requirement, thereby being highly advantageous in real-world scenarios when compared to Hyper and Spectral.

To summarize, our extensive experimental evaluation portrays the *efficiency, scalability, and effectiveness* of OverlapH in solving the CDR-Multi problem. Additionally, its ability to perform data and replica placement in a single step allows for a better and unified system design.

VII. CONCLUSION AND FUTURE WORK

The problem of *combined data and replica placement* has been addressed in this article. Although replication is an integral part of data placement, we identified that most of the techniques in the literature do not address the two placement steps as a single joint optimization problem, but rather treat them as two independent problems. Hence, existing techniques employed a two-stage approach: performing data placement followed by replica placement. Consequently, with the objective of *combining data and replica placement* we proposed a unified paradigm, called *CDR*, which facilitates the two problems to be studied jointly. We proposed two variants of the CDR problem: *CDR-Single* and *CDR-Multi* with applicability in addressing use-cases under two interesting real-world application domains: OLAP and OSN, respectively. To effectively solve the CDR problem (and its variants), we proposed a generic framework, called *UnifyDR*, which possessed the capability to unify data and replica placement. We also proposed two algorithms, namely – *OverlapG* and *OverlapH*, possessing the capability of partitioning a set of data-items by allowing each data-item to be assigned to multiple nodes, thereby facilitating joint optimization of data and replica placement. While OverlapG performed overlapping correlation clustering on a graph to address the CDR-Single problem, the OverlapH algorithm performed overlapping clustering on hypergraphs for solving the CDR-Multi problem. To evaluate the effectiveness and efficiency of the proposed algorithms OverlapG and OverlapH, we performed experiments on data simulated using a decision support benchmark and a trace-based social network dataset respectively. It was identified that the proposed algorithms are approximately 20–30% better on the evaluated metrics while being 2–8 times faster.

Currently, UnifyDR and its algorithms (OverlapG and OverlapH) perform a static analysis of the considered workload to learn a data and replica placement strategy. In other words, they lack the ability to manage dynamic workloads. In the future, the focus will be to make UnifyDR and the underlying algorithms capable of handling updates in the data in an online manner, and dynamically updating the CDR placement output.

REFERENCES

[1] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," in *Proc. NSDI*, 2010, pp. 1–16.

[2] Amazon. (2017). *AWS Global Infrastructure*. Accessed: Aug. 10, 2020. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/

[3] A. Atrey, G. Van Seghbroeck, H. Mora, F. De Turck, and B. Volckaert, "SpeCH: A scalable framework for data placement of data-intensive services in geo-distributed clouds," *J. Netw. Comput. Appl.*, vol. 142, pp. 1–14, Sep. 2019.

[4] A. Atrey, G. Van Seghbroeck, H. Mora, F. D. Turck, and B. Volckaert, "Unifying data and replica placement for data-intensive services in geographically distributed clouds," in *Proc. 9th Int. Conf. Cloud Comput. Services Sci.*, 2019, pp. 25–36.

[5] A. Atrey, G. Van Seghbroeck, B. Volckaert, and F. D. Turck, "Scalable data placement of data-intensive services in geo-distributed clouds," in *Proc. 8th Int. Conf. Cloud Comput. Services Sci.*, 2018, pp. 497–508.
[6] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping correlation clustering," *Knowl. Inf. Syst.*, vol. 35, no. 1, pp. 1–32, Apr. 2013.
[7] U. V. Catalyurek. (2011). *PaToH (Partitioning Tool for Hypergraphs)*. Accessed: Aug. 10, 2020. [Online]. Available: http://bmi.osu.edu/umit/PaToH/manual.pdf
[8] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Rec.*, vol. 26, no. 1, pp. 65–74, Mar. 1997.
[9] A. Chervenak, E. Deelman, M. Livny, M.-H. Su, R. Schuler, S. Bharathi, G. Mehta, and K. Vahi, "Data placement for scientific applications in distributed environments," in *Proc. 8th IEEE/ACM Int. Conf. Grid Comput.*, Sep. 2007, pp. 267–274.
[10] Cisco. (2018). *Cisco Visual Networking Index: Forecast and Trends (2017–2022)*. Accessed: Aug. 10, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
[11] T. P. P. Council. (2006). *The TPC-DS Benchmark*. Accessed: Aug. 10, 2020. [Online]. Available: http://www.tpc.org/tpcds/
[12] C. Curino, E. Jones, Y. Zhang, and S. Madden, "Schism: A workload-driven approach to database replication and partitioning," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 48–57, Sep. 2010.
[13] Y. Ding and Y. Lu, "Automatic data placement and replication in grids," in *Proc. Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2009, pp. 30–39.
[14] M. Ebrahimi, A. Mohan, A. Kashlev, and S. Lu, "BDAP: A big data placement strategy for cloud-based scientific workflows," in *Proc. IEEE 1st Int. Conf. Big Data Comput. Service Appl.*, Mar. 2015, pp. 105–114.
[15] M. Y. Eltabakh, Y. Tian, F. Özcan, R. Gemulla, A. Krettek, and J. McPherson, "CoHadoop: Flexible data placement and its exploitation in Hadoop," *Proc. VLDB Endowment*, vol. 4, no. 9, pp. 575–585, 2011.
[16] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "An algorithm for network and data-aware placement of multi-tier applications in cloud data centers," *J. Netw. Comput. Appl.*, vol. 98, pp. 65–83, Nov. 2017.
[17] L. Golab, M. Hadjieleftheriou, H. Karloff, and B. Saha, "Distributed data placement to minimize communication costs via graph partitioning," in *Proc. 26th Int. Conf. Sci. Stat. Database Manage. (SSDBM)*, 2014, pp. 1–12.
[18] R. K. Grace and R. Manimegalai, "Dynamic replica placement and selection strategies in data grids—A comprehensive survey," *J. Parallel Distrib. Comput.*, vol. 74, no. 2, pp. 2099–2108, 2014.
[19] W. Guo and X. Wang, "A data placement strategy based on genetic algorithm in cloud computing platform," in *Proc. 10th Web Inf. Syst. Appl. Conf.*, Nov. 2013, pp. 369–372.
[20] A. Hadoop. (2018). *HDFS Architecture Guide*. Accessed: Aug. 10, 2020. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#References
[21] S. Han, B. Kim, J. Han, K. Kim, and J. Song, "Adaptive data placement for improving performance of online social network services in a multicloud environment," *Sci. Program.*, vol. 2017, pp. 1–17, Aug. 2017.
[22] K. Huguenin, A.-M. Kermarrec, K. Kloudas, and F. Taïani, "Content and geographical locality in user-generated content sharing systems," in *Proc. 22nd Int. Workshop Netw. Operating Syst. Support Digit. Audio Video (NOSSDAV)*, 2012, pp. 77–82.
[23] Inside Bigdata Editorial Team. (2017). *The Exponential Growth of Data*. Accessed: Aug. 10, 2020. [Online]. Available: https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data/
[24] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 28–36.
[25] G. Karypis and V. Kumar. (1995). *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*. Accessed: Aug. 10, 2020. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
[26] T. Kosar and M. Livny, "Stork: Making data placement a first class citizen in the grid," in *Proc. 24th Int. Conf. Distrib. Comput. Syst.*, Mar. 2004, pp. 342–349.
[27] T. Kosar and M. Livny, "A framework for reliable and efficient data placement in distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 65, no. 10, pp. 1146–1157, Oct. 2005.
[28] K. A. Kumar, A. Deshpande, and S. Khuller, "Data placement and replica selection for improving co-location in distributed environments," *CoRR*, vol. abs/1302.4168, 2013. [Online]. Available: https://arxiv.org/abs/1302.4168
[29] P. Li, H. Dau, G. Puleo, and O. Milenkovic, "Motif clustering and overlapping clustering for social network analysis," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.
[30] X. Li, L. Zhang, Y. Wu, X. Liu, E. Zhu, H. Yi, F. Wang, C. Zhang, and Y. Yang, "A novel workflow-level data placement strategy for data-sharing scientific cloud workflows," *IEEE Trans. Services Comput.*, vol. 12, no. 3, pp. 370–383, May/Jun. 2019.
[31] X. Liu and A. Datta, "Towards intelligent data placement for scientific workflows in collaborative cloud environment," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, May 2011, pp. 1052–1061.
[32] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. NIPS*, 2001, pp. 849–856.
[33] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling memcache at Facebook," in *Proc. NSDI*, 2013, pp. 385–398.
[34] A. Quamar, K. A. Kumar, and A. Deshpande, "SWORD: Scalable workload-aware data placement for transactional workloads," in *Proc. 16th Int. Conf. Extending Database Technol. (EDBT)*, 2013, pp. 430–441.
[35] Y. Rochman, H. Levy, and E. Brosh, "Resource placement and assignment in distributed network topologies," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1914–1922.
[36] T. P. Shabeera, S. D. M. Kumar, S. M. Salam, and K. M. Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm," *Eng. Sci. Technol., Int. J.*, vol. 20, no. 2, pp. 616–628, Apr. 2017.
[37] P. N. Shankaranarayanan, A. Sivakumar, S. Rao, and M. Tawarmalani, "Performance sensitive replication in geo-distributed cloud datastores," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 240–251.
[38] Z. Wang. (2017). *Latency Between AWS Global Regions*. Accessed: Aug. 10, 2020. [Online]. Available: http://zhiguang.me/2016/05/10/latency-between-aws-global-regions/
[39] T. White, *Hadoop: The Definitive Guide*. Newton, MA, USA: O'Reilly Media, Inc., 2012.
[40] Wikipedia. (2018). *Laplacian Matrix*. Accessed: Aug. 10, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Laplacian_matrix
[41] B. Yu and J. Pan, "Location-aware associated data placement for geo-distributed data-intensive applications," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 603–611.
[42] B. Yu and J. Pan, "Sketch-based data placement among geo-distributed datacenters for cloud storages," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[43] B. Yu and J. Pan, "A framework of hypergraph-based data placement among geo-distributed datacenters," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 395–409, May/Jun. 2020.
[44] T. Yu, J. Qiu, B. Reinwald, L. Zhi, Q. Wang, and N. Wang, "Intelligent database placement in cloud environment," in *Proc. IEEE 19th Int. Conf. Web Services*, Jun. 2012, pp. 544–551.
[45] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1200–1214, Oct. 2010.
[46] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.
[47] J. Zhang, J. Chen, J. Luo, and A. Song, "Efficient location-aware data placement for data-intensive applications in geo-distributed scientific data centers," *Tsinghua Sci. Technol.*, vol. 21, no. 5, pp. 471–481, Oct. 2016.
[48] Q. Zhao, C. Xiong, and P. Wang, "Heuristic data placement for data-intensive applications in heterogeneous cloud," *J. Electr. Comput. Eng.*, vol. 2016, pp. 1–8, May 2016.
[49] Q. Zhao, C. Xiong, K. Zhang, Y. Yue, and J. Yang, "A data placement algorithm for data intensive applications in cloud," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 2, pp. 145–156, Feb. 2016.
[50] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. NIPS*, 2006, pp. 1601–1608.

ANKITA ATREY received the master's degree in computer science from the Vellore Institute of Technology (VIT), Vellore, India. She is currently pursuing the Ph.D. degree with the Department of Information Technology (INTEC), Ghent University, Belgium, and imec. She has internship experience from the CNRS, France, and the Indian Institute of Technology (IIT) Kanpur, India. Her research interests include cloud computing: resource scheduling and provisioning, data-placement, service management, and service oriented architectures. At INTEC, she is also working on research problems encircling intelligent resource provisioning in multi-tenant multicomponent applications. She has published her research in reputed cloud and service management journals and conferences, such as Springer JNCA, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), CNSM, SC2, and CLOSER, while also serving as a Reviewer for CLOSER, *Journal of Network and Systems Management* (JONS) and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM).

GREGORY VAN SEGHBROECK graduated from Ghent University, in 2005. He received the Ph.D. degree in computer science engineering in 2011. After a brief stop as an IT Consultant, he joined the Department of Information Technology (INTEC), Ghent University, (currently IDLab). In January 2007, he received the Ph.D. degree grant from IWT, Institute for the Support of Innovation through Science and Technology, to work on theoretical aspects of advanced validation mechanism for distributed interaction protocols and service choreographies. His main research interests focus on big data engineering and complex scalable cloud platforms.

HIGINIO MORA received the B.S. degree in computer science engineering and the B.S. degree in business studies from the University of Alicante, Spain, in 1996 and 1997, respectively, and the Ph.D. degree in computer science from the University of Alicante in 2003. Since 2002, he has a member of the faculty of the Computer Technology and Computation Department, University of Alicante, where he is currently an Associate Professor and a Researcher with the Specialized Processors Architecture Laboratory. His research interests include computer modeling, computer architectures, high-performance computing, embedded systems, the Internet of Things, and cloud computing paradigm. He has participated in many conferences and most of his work has been published in international journals and conferences, with more than 50 published articles.

BRUNO VOLCKAERT (Member, IEEE) received the master's degree in computer science from Ghent University, in 2001, and the Ph.D. degree in data intensive scheduling and service management for grid computing from Ghent University, in 2006. He is currently a Professor of advanced programming and software engineering with the Department of Information Technology (INTEC), Ghent University, and a Senior Researcher with imec. His current research interests include with reliable and high-performance distributed software systems for City-of-Things (IoT for Smart Cities), distributed decision support systems for UAVs, intelligent railway transportation applications, and autonomous optimization of cloud-based applications. He has worked on more than 35 national and international research projects and is author or coauthor of more than 80 papers published in international journals and conference proceedings.

FILIP DE TURCK (Senior Member, IEEE) currently leads the Network and Service Management Research Group, Department of Information Technology, Ghent University, Belgium, and imec. He has (co)authored more than 450 peer-reviewed articles. His research interests include telecommunication network and service management, efficient big data processing, and the design of large-scale the Internet of Things systems. In this research area, he is involved in several research projects with industry and academia, serves as the Vice-Chair for the IEEE Technical Committee on Network Operations and Management (CNOM), the Chair for the Future Internet Cluster of the European Commission, and is on the TPC of many network and service management conferences and workshops, and serves on the Editorial Board of several network and service management journals. Specifically, he serves as the Editor-in-Chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), and as a Steering Committee Member of the IEEE Conference on Network Softwarization (IEEE Net-Soft).