International Journal of Scientific Research in Computer Science, Engineering and Information Technology
 | Volume 4 | Issue 2 | ISSN : 2456-3307

Wide Area Analytics: Efficient Analytics for a Geo -Distributed Datacenters

Alapati Janardhana Rao [*1], Bellamkonda Naresh [2]

[1*]janardhana182@gmail.com , nareshbellamkonda7075@gmail.com

ABSTRACT

Large organizations today operate data centers around the globe where massive amounts of data are produced and consumed by local users. Despite their geographically diverse origin, such data must be analyzed/mined as a whole. We call the problem of supporting rich DAGs of computation across geographically distributed data Wide-Area Big-Data (WABD). To the best of our knowledge, WABD is not sup-ported by currently deployed systems nor sufficiently studied in literature; it is addressed today by continuously copying raw data to a central location for analysis. We observe from production workloads that WABD is important for large organizations, and that centralized solutions incur substantial cross-data center network costs. We argue that these trends will only worsen as the gap between data volumes and transoceanic bandwidth widens. Further, emerging concerns over data sovereignty and privacy may trigger government regulations that can threaten the very viability of centralized solutions. To address WABD we propose WANalytics, a system that pushes computation to edge data centers, automatically optimizing work own execution plans and replicating data when needed. Our Hardtop-based prototype delivers 257 reductions in WAN bandwidth on a production workload from Microsoft. We round out our evaluation by also demonstrating substantial gains for three standard benchmarks: TPC-CH, Berkeley Big Data, and Big Bench.

Keywords: Big Data, Analytics, Geo-Distributed Datacenters

I. INTRODUCTION

Many large organizations today have a planetary-scale footprint and operate tens of data centers around the globe. Local data centers ensure low-latency access to users, avail-ability, and regulatory compliance. Massive amounts of data are produced in each data center by logging interactions with users, monitoring compute infrastructures, and tracking business-critical functions. Analyzing all such data glob-ally in a consistent and unified manner provides invaluable insight. We refer to the problem of supporting arbitrary

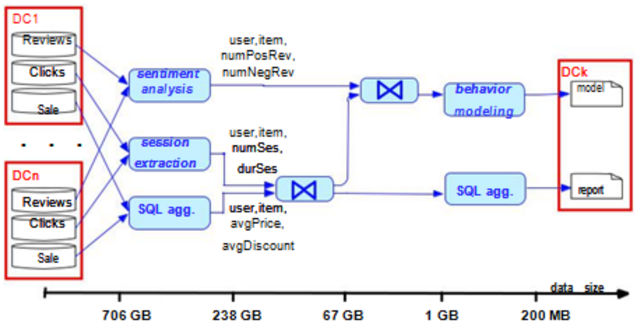

Figure 1. Running example

DAGs of computation over data born in geographically dis-tributed data centers as Wide-Area Big Data (WABD), and argue for solutions that are cross-data center bandwidth conscious.

We introduce WABD using the running example in Figure 1. This example, derived from Big Bench [21], is representative of the data processing needs we observe in Microsoft production Workloads. It is also consistent with data processing pipelines at Yahoo! [20], Face book [38], Twitter [29], and LinkedIn [12]. Figure 1 shows three input data sources: click stream, storing Web server logs of user browsing activities; reviews, capturing textual representations of item reviews; and sales, a relational table storing transactional records of item purchases. Unlike parallel databases and Big Data systems, the data is distributed across \nodes" (i.e., data centers) to reduce latency of Web server interactions and not to scale-out the analytical framework. As a result, we have no control on the base data partitioning: data is born distributed; we only control data replication and distributed execution strategies. The DAG of operators shown in Figure 1 depicts one of the many work owes run daily to process the raw data and extract insight about user behavior, sales performance, and item reception. In particular, beside classical relational operators, this work own includes arbitrary computations that manipulate unstructured data (session extraction and sentiment analysis) and ma-chine learning stages (behavior modeling).

The practical relevance of data analysis like this one can be seen in the dozens of single- and multi-machine relational Online Analytical Processing (OLAP) systems [6, 7, 4, 10, 14], and more recently with the development of massively scalable data processing frameworks [36, 39, 3, 40, 34], collectively referred to here as Big Data systems. All these Systems provide sophisticated single-cluster analytics solutions. Recent e orts [23] have focused on data replication for disaster recovery, but their analytics components still operate on a single data center. We discuss the vast related work in distributed databases and work own management systems in x4.

Companies deal with WABD today by copying remotely-born data to a central location for analysis. Throughout the paper we will refer to this as the centralized solution. Any such solution is destined to consume cross-data center bandwidth proportional to the volume of updates/growth of the base data. Referring back to Figure 1, this consists of copying the partitions for the three base data sources click-stream, reviews, and sales, from the edge data centers to a central location, and running the DAG leveraging standard single-cluster technologies. For example, using a Hardtop stack, one could use DistCP to copy data across HDFS in-stances in each data center, Oozie to orchestrate the workflow, Hive for relational processing, MapReduce for session extraction, OpenNLP for sentiment analysis, and Mahout for behavior modeling. We prototyped this setup and gathered initial numbers to quantify the cost of this approach. Assuming daily runs of the DAG of Figure 1, 1 TB daily data growth, and 10 data centers, we observe cross data center traffic of 706 GB per day. (Other base data sources in the original benchmark, not used by the DAG in Figure 1, make up another 318 GB per day.)

The distributed solution we propose significantly reduces this bandwidth cost. Further, we argue that any centralized solution is on a collision course with current technological and geo-political trends: 1) data is growing much faster than cross-data center bandwidth, and 2) growing privacy and data sovereignty concerns are likely to result in government regulation that threatens the viability of centralized approaches (e.g., German-born data might not be stored in US data centers). The table below summarizes how WABD differs from classical database problems.

Table 1

	WABD Problem: Novel dimensions
Data Placement	No control on data partitioning (only on replication)
Target Computation	Arbitrary DAGs (vs relational)
Optimization Metrics	Cross-data center bandwidth (abundant CPU/Storage)
New Constraints	Data sovereignty + Heterogeneous bandwidth

To address the challenges of WABD, we build WANalytics, a system that supports arbitrary DAGs of computation on geographically distributed data. WANalytics automatically devises distributed execution plans and an accompanying data replication strategy. These two aspects are ad-dressed concurrently to minimize WAN bandwidth utilization while respecting regulatory requirements. In designing WANalytics we make three contributions:

1. We introduce a caching mechanism akin (for networking) to syntactic view maintenance for arbitrary computations.
2. We explore the joint space of distributed execution plans and data replication strategies, and propose a greedy heuristic, and show its limitations.
3. We propose \pseudo-distributed measurement", a technique that circumvents cardinality estimation by running user queries as if they were distributed and measuring their actual cost.

WANalytics vastly outperforms the centralized approach, moving only 1.07 GB across data centers when tasked with the DAG of our running example. We obtain similar re-

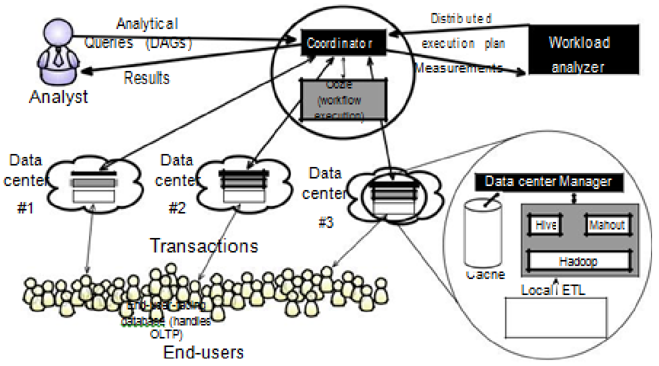

Figure 2. WANalytics architecture

sults (over 250 bandwidth reduction) on a large production workload from Microsoft, on TPC-CH [16], and Big-Bench [21] benchmarks, and show more modest gains for the Berkeley Big Data Benchmark [13] (x3).

We conclude by discussing how WANalytics can serve as a starting point to address data sovereignty concerns, and by listing several open questions regarding approximate query answering, differential privacy, query optimization, view selection and incremental maintenance for the emerging prob-lem of Wide-Area Big Data.

II. WANALYTICS OVERVIEW

WANalytics pushes computation to edge data centers, and replicates the base data when needed, while respecting data sovereignty constraints. We describe the architecture, the components, and two new techniques (pseudo-distributed measurement and syntactic view maintenance) that underly our system.

2.1 Architecture

Analytics' consists of two main components (see Figure 2): (1) a runtime layer that executes user DAGs in a distributed way across data centers, moving data according to the execution and data replication strategies devised by (2) a workload analyzer that continuously monitors and optimizes the user workload.

Runtime layer comprises a centralized coordinator in a master data center that interacts with data center managers at each edge data center. Users submit logical DAGs of computation (such as the one in Figure 1) to the coordinator. The coordinator in turn asks the workload analyzer to provide a physical distributed execution plan for a given DAG. The physical plan explicitly specifies where each stage is going to be executed, and how data will be transferred across data centers. We leverage Apache Oozie [1] to handle the mechanics of orchestrating distributed execution, and simple faults, while we provide purpose-built components to collect statistics on job execution and base data growth (e.g., volume of updates on the base data). By design our system can support DAG operators expressed in any frame-work compatible with Oozie (e.g., Hive, Pig, MapReduce, Spark). After execution completes, job statistics from each data center manager are sent to the workload analyzer to aid future query optimization. As part of our runtime layer we developed a data transfer optimization that minimizes the redundancy of subsequent transfers (x2.2).

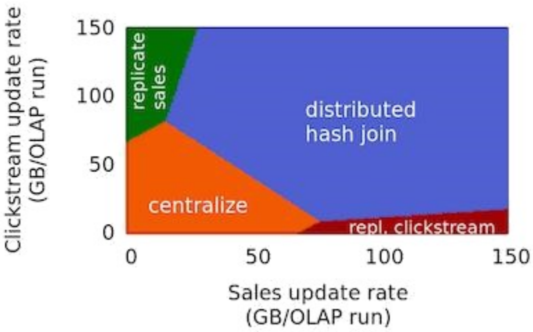

Figure 3. Optimal strategy for session extraction. / Sales summarization

Workload analyzer WANalytics is targeted towards applications with a slowly-evolving core of recurring jobs (DAGs) that make up the bulk of the workload. This is a reason-able assumption in the context of WABD, as we confirm by inspecting several production workloads at Microsoft. The workload analyzer jointly optimizes all the DAG execution plans of the workload, along with the data replication policy. The resulting search space is inherently vast, and we thus propose a greedy heuristic to explore it efficiently in x2.3. During this optimization step the system translates the logical input DAGs into fully qualified distributed physical plans. We take a two-step approach, in which we first generate an optimized centralized plan and then add distribution to it, as often done in the past [27]. We handle conservatively all operators with unknown semantics/requirements, i.e., we assume they can only be run in a single data center on local data. As an example consider the machine learning step of behavior modeling in our running example. On the other hand, we leverage the known semantics of relational operators and Map Reduce jobs when possible. For example we detect that the sentiment analysis stage in our running example is a map-only job, and we can thus \push-down" its execution to each edge data center. Figure 4 shows the results of this optimization for our running example.

The analyzer costs each alternative execution by means of the technique we discuss in x2.4. The workload analyzer also establishes a policy for base data replication, i.e., it decides whether for each base data source it is worthwhile to maintain a replica in another data center to reduce network transfers.

The workload analyzer is run every epoch (e.g., once a day). This allows us to continuously assess the performance of current strategies and investigate alternative options. The goal is to progressively improve upon the current strategies by carefully exploring the space. Each change of strategy is carefully vetted (x2.4), and we favor robust plans, since the cost of mistakes in our environment is very high.

We next discuss: an optimization we deploy to reduce the cost of data transfer (x2.2); the algorithm used by our proto-type of the workload analyzer (x2.3); and the technique we use to make it possible to cost alternative execution strategies (x2.4).

2.2 Data transfer optimization

The unique setting we consider, in which each \node" is a full data center with virtually limitless CPU and storage capabilities, and connectivity among nodes is very costly/limited, lends itself to a novel optimization. We cache all intermediate results generated during DAG execution at each data center, and systematically compute di s[1] to re-duce cross-data center bandwidth. Whenever a source data center S sends the result for a computation C to a destina-tion data center D, both S and D store a copy of the result of C in a cache tagged with hsignature(C)i[2]. The next time S needs to send results for C to D, it evaluates C again, but instead of sending the result afresh, it computes a di between the old and new result, and sends the smaller be-tween the di and new result. D then applies the di onto its copy of the old result.

In many cases a change in the base data a ects only part of the output of C, hence a signi cant data transfer bene-t can be obtained by our data transfer optimization. Our approach is agnostic of what C does, but systematically re-moves redundant data transfers, by detecting overlap. This is done at the cost of increased computation and storage re-quirements on each data center (to cache data and compute di s).

Interestingly, this form of caching helps not only when end-users submit the same DAG repeatedly, but also by eliminating redundant transfers across DAGs sharing com-mon sub-computations. In the TPC-CH benchmark, 6 dif-ferent DAGs all compute slightly di erent aggregates on top of the same (relatively data-intensive) join. Caching reduces the data transfer for these DAGs by 5:99 .

In a sense, our data transfer optimization is a syntactic form of view maintenance for arbitrary computations. We materialize an implicit view the rst time a computation ar-rives, and lazily refresh it at every subsequent query that overlaps this. The purely syntactic nature of this process allows it to function for arbitrary computation. However, compared to classical relational view maintenance mecha-nisms, our cache is likely to waste computation/storage, and possibly miss opportunities for optimization.

2.3 Workload analyzer

The workload analyzer takes as input a set of logical DAGs and the base data natural partitioning. It then determines the combination of choices for the following three factors that would minimize total bandwidth usage: (1) the phys-ical operator to use for tasks that accept multiple imple-mentations (e.g., hash, broadcast or semi join), (2) the data center where each task is executed (respecting sovereignty constraints), and (3) the set of data centers to which each partition of the base data is replicated. These decisions are di cult for several reasons.

First, nding the best execution strategy for each task in a DAG in isolation, is by itself non-trivial. For example, the choice of optimal join execution strategy is a complicated function of several parameters: the sizes of the base tables, the rates at which they are updated, the selectivities of each of the task in the DAG, etc. Figure 3 shows the best join strategy for the join between the output of the session ex- Di s are computed at the record level, if the record format is known, or on the binary representation otherwise.

[2]signature(C) = depth- rst traversal of the sub-DAG in-duced by C. This mechanism is imperfect { e.g. changing the order in which DAG edges are listed can change the sig-nature and cause a cache miss { but is a reasonable starting point.

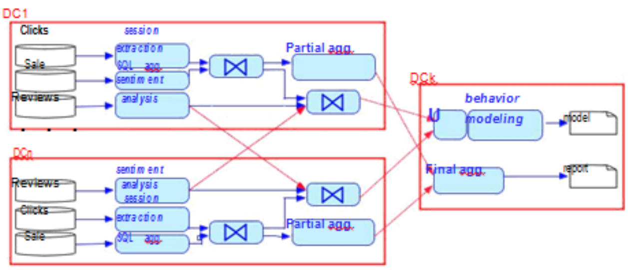

Figure 4. Placement of the running example DAG

traction and sales summarization tasks from Figure 1, as a function of the rate at which the sales and the click-stream tables are updated. Depending on the update rates of the base tables, one of the following strategies dominates: copying both tables centrally, broadcasting the updates of the least modi ed table, or performing a distributed hash join (i.e., re-distributed both tables via hashing). Second, the choice of execution strategy for a DAG node may a ect the choice of strategies for other DAG nodes, as this choice determines the partitioning and placement of the node's out-put data. In a workload with n nodes (in one or more DAGs) and up to k possible execution strategies per node, the an-alyzer would have to explore a $O(k^n)$ search space. Third, the choices made for nodes of di erent DAGs in uence each other, as they might leverage a shared data replication strat-egy, or be a ected by the optimization discussed in x2.2. Hence, we propose a Greedy Heuristic that performs remark-ably well in practice, while exploring only a small subset of the search space.

Algorithm 1 Greedy Heuristic

for all DAG G 2 workload do
  for all task t 2 toposort(G) do
    t.completed = false
    if 9 parent p of t such that p.completed =
      false then assign a default strategy to t
    else
      if all strategies for t have been
        evaluated then for all data source
        S 2 input(t) do
          test if replicating S reduces bandwidth
        further assign the lowest cost strategy to
        t, and replication
strategy to S
        t.completed = true
      else
        explore next strategy for t

Greedy Heuristic The heuristic (Algorithm 1) optimizes each node of each DAG in isolation, proceeding from the source nodes and moving greedily outward in topological order. For each node, we evaluate all strategies compatible with sovereignty constraints, using pseudo-distributed mea-surement (x2.4) to measure their costs, and greedily pick the lowest cost alternative at that node. In the process, the sys-tem also evaluates whether systematically replicating any of the input base tables can help amortize transfer costs among DAGs.

Figure 4 shows the resulting execution strategy for the DAG in our running example. The arrows in red are cross-data center data transfers, and add up to 1.07 GB. Most of the cost is incurred while broadcasting the output of senti-ment analysis during join computation. The alternatives{ such as using a semi-join, or redistributing via hashing| all turn out to be more expensive. In our running example,

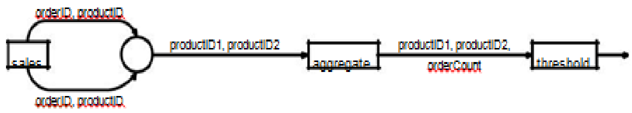

Figure 5. Example where heuristic fails

WANalytics decides not to replicate base tables, but replica-tion proves fundamental for all workloads in our experiments (x3).

This simple heuristic requires a limited number of mea-surements (as it explores just a small portion of the search space), and experimentally works well whenever DAGs \re-duce" data volumes at each subsequent stage. This seems common in practice: it is true of 98% of all the DAGs in our workloads.

Limitations This heuristic can fail when confronted with DAGs that \expand" the input data they consume (before optionally condensing it). Consider the DAG in Figure 5, a simpli ed version of query Q1 in the BigBench bench-mark. The DAG starts from a table listing items ordered by customers (size n), performs a self-join on the table to nd pairs of items that are ordered together (worst case size $O(n^2)$), computes frequencies of pairs, and returns frequent pairs. The heuristic would push the join down and run it distributed, thus exploding data in edge data centers, in-curring unnecessarily large data transfer during the second stage.

We are actively working on a non-linear integer program-ming (NLIP) model that can explore the search space more systematically. We currently have a limited IP formulation for the special cases when either all nodes in the DAG are SQL operators or all the nodes are MapReduce operations it turns out that this formulation does identify the correct strategy in this example. However, at the time of this writ-ing, the arbitrary DAGs we allow in our system are beyond our reach.

2.4 Pseudo-distributed measurement

Like most optimizers, our workload analyzer must cost each strategy it considers. Traditional cardinality estima-tion techniques, based on data statistics and histograms, are insu cient for the arbitrary computations we target. We propose pseudo-distributed measurement, a mechanism that allows us to \measure" the cost of running a DAG as if it was executed according to a di erent strategy.

Consider the DAG in Figure 1, and assume the system is currently deployed in a centralized manner (i.e., all data are replicated centrally), but we want to explore the cost of a distributed execution (e.g., one in which session extrac-tion is run on each data center). The analyzer would need to estimate data volumes produced by each operator when run on the portion of the input data stored at each data center. This information is not directly available when the query is run centralized (as input data are combined during operator execution and their provenance is lost).

To estimate data sizes when multiple (input or interme-diate) data partitions are housed in a single physical data center, WANalytics simulates a virtual data center topol-ogy in which each partition of input data is in a separate data center. This is done by decomposing an operator O in the DAG into multiple sub-operators O_i, each executing on a portion of the data and a nal stage O_c combining their results { this is done only for operators whose semantics is compatible with such decomposition. We enrich the data with a \provenance" eld that tracks the data centers where they were born. This allows O_i to lter the data and run only on data coming from data center i. We then measure the size of each O_i output and infer the cost of running O dis-tributed. In the session extraction example, WANalytics would create [# of DCs] separate tasks that operate on each partition of the clickstream dataset separately, computing session statistics for each user at each data center, followed by a simulated data transfer to a virtual central node that computes the nal aggregate session statistics for each user.

Pseudo-distributed measurement does a ect execution performance (up to 20% in our experiments). This can be mitigated by increasing parallelism of execution in most cases (not our focus here). Most importantly, each measure-ment runs within a single data center, thus never increases the volume of data transferred between data centers. We omit the details of how the measurements are executed when the execution plan is already distributed.

To summarize, pseudo-distributed measurement consists of a costly but very accurate measurement of the execution costs under di erent execution strategies, achieved by means of DAG rewriting and actual execution.

Limitations To explore all options considered by the heuristics of Section 2.3, WANalytics has two options: (1) run a di erent pseudo-distributed measurement every time the DAG is submitted by the user, or (2) run dedicated jobs that have the sole purpose of costing di erent options. In either case, if d is the longest path of any of the DAGs in the workload, and k the maximum number of strategies available for any task, we might require up to k d epochs/submissions to collect all needed measurements.

This could be slow in certain settings, and indicates that hybrid solutions, combining the low-cost of classical cardinality estimation using statistics/histograms with the high-precision of pseudo-distributed measurement, are likely needed.

III. EXPERIMENTAL EVALUATION

In this section, we compare WANalytics with a centralized deployment over various workloads.

3.1 Workloads and Experimental Setup

Microsoft production workload: This use case con-sists of a monitoring infrastructure collecting tens of TBs of service health/telemetry data daily at geographically dis-tributed data centers. The data are continuously replicated to a central location, and analyzed by means of a mix of purpose-built near-real time analysis and Hive-based batch analytics. The bulk of the load is produced by few tens of canned DAGs producing aggregate reports on service utiliza-tion and infrastructure health, and by thousands of ad-hoc analytical queries (submitted over a 6 month period of time by the engineering team for triaging/testing purposes). The complexity of the DAGs ranges from small to medium (up to tens of stages).

BigBench: This workload [21] has been recently proposed as a rst step towards a SPEC standardization. It comprises a relational portion, TPC-DS, and the two non-relational sources we show in our running example, clickstream and reviews. In our multi-data center setup, we assume data are

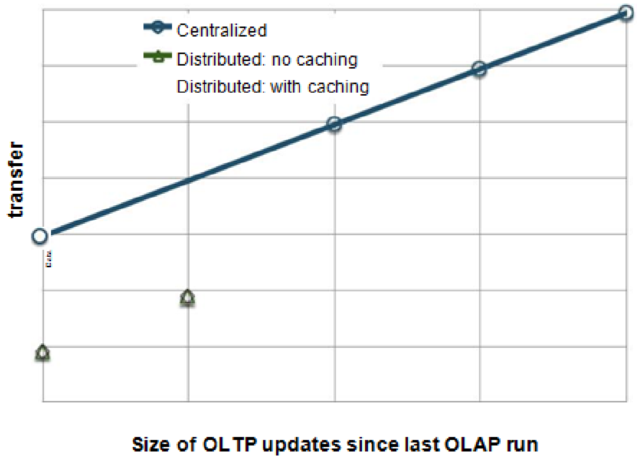

Figure 6. WANalytics vs centralized baseline on Mi-crosoft production workload

produced at edge data centers, and that accesses by the same user are logged in one location. This reasonable assumption of co-location lowers the cost of joins, but is not required for WANalytics to work correctly.

TPC-CH: The TPC-CH benchmark [16] models the database for a large-scale product retailer such as Amazon, and is a joint OLTP/OLAP benchmark, constructed by combining the well-known TPC-C OLTP benchmark with the TPC-H OLAP benchmark. In our experiments, we assume that the TPC-C portion of the workload is exercised locally at each edge data center, while the TPC-H analytics are meant to be run on the overall data.

Berkeley BigData Benchmark: This benchmark [13], developed by the AMPLab at UC Berkeley, models a database generated from HTTP server logs[3]. Once again we assume that the raw data are logged by web servers at edge data centers.

Experimental setup. We ran our experiments across three geographically distributed Azure data centers (US, EU, Asia), and on a large on-premise cluster, on which we sim-ulate a multi-data center setup. Speci cally, we ran the benchmark-based experiments in both deployments for up to 25 GB of data transfers. Experiments on the larger Mi-crosoft workload and on benchmarks in the 25 GB to 10 TB range have been conducted exclusively on the on-premise cluster. This is because each of the multi-terabyte runs for the baseline centralized approach would have otherwise cost thousands of dollars in cross-data center bandwidth. The on-premise cluster consists of 120 machines, each with 128 GB of RAM, 32 cores, and 12 x 3 TB of drives. The inter-connect is 10 Gbps within a rack, and 6 Gbps across any two machines.

3.2 Comparing WANalytics with Centralized

The software stack we use in these experiments is based on a combination of Oozie, Hive, MapReduce, OpenNLP, Ma-hout and DistCP. Since our focus is on network bandwidth consumption and not query execution performance, we ex-pect similar results from alternative choices of stack. All network transfers, both baseline and WANalytics, are gzip-compressed. For the centralized baseline we always pick the best between log-shipping and batch-copying.

Figure 6 shows the results of running WANalytics and the centralized baseline on the Microsoft workload (axes hidden due to proprietary nature of the data). Figure 7 shows an equivalent set of experiments for the three standard bench-marks we consider. On the y-axis we report the cross-data center network bandwidth consumed by each approach, for di erent volumes of update/growth of the base data be-tween runs of the analytical workload. This is consistent with our observation of production workloads, where anal-ysis is run on a x daily schedule, while the raw volume of data growth/update changes with the service popularity (in this case, growing aggressively over time). Our workload an-alyzer consistently picks the lowest among centralized and distributed solutions from Figures 6, 7. To avoid crowding the gures, we omit these lines. The key insights from these experiments are:

1. The centralized approach grows linearly with raw data updates/growth. Note that slope is < 1 due to com-pression.
2. Controlling base-table replication is key to lower bandwidth consumption for frequently-read, rarely-updated tables (e.g., dimension tables in TPC-CH).
3. At low update rates, centralized outperforms dis-tributed for two of the four workloads. This is because frequent analytics operate on mostly unchanged data.
4. At high update rates distributed outperforms centralized by 3 to 360 . The larger advantages accrue for workloads where we can push operators to edge data centers more e ectively. The Berkeley Big-Data Benchmark results are dominated by a single query, which requires to move large amounts of data to compute a top-k.
5. At low/medium update rates caching is e ective. This is due to the large redundancy among the answers to subsequent runs of overlapping queries. By contrast, at high-update rates, and for queries with no overlap (Fig. 6) caching is not e ective, since transfer redundancy is al-ready minimal.

Overall, these results are very encouraging, and con rm a substantial opportunity to address WABD by means of distributed execution of complex DAGs.

IV. CONCLUSION

As data develops at a gigantic rate, accomplishing ideal execution in the wide region investigation turns out to be increasingly challengeable. Contrasted and the neighborhood organize in a datacenter, the WAN covers a generally expansive land region, which is more convoluted and insecure. In addition, handling a considerable measure of data inside a little time interim is an incredible test for those low dormancy cloud applications. In this paper, we exhibit various run of the mill components in the wide region investigation, examine abnormal state thoughts, and give a correlation of these systems. Despite the fact that with a few constraints, more viable arrangements might be enlivened by these components and connected in reality soon.

V. REFERENCES

[1]. Apache Oozie. http://oozie.apache.org.

[2]. Apache Storm. http://storm.incubator.apache.org.

[3]. Cloudera Impala. http://bit.ly/1eRUDeA.

[4]. Greenplum. http://bit.ly/1oL4Srq.

[5]. Greenplum. http://basho.com/riak/.

[6]. Netezza. http://www.ibm.com/software/data/netezza/.

[7]. Vertica. http://www.vertica.com/.

[8]. S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu,I. Stoica, and J. Zhou. Reoptimizing data parallel computing. In Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pages 281{294, San Jose, CA, 2012. USENIX.

[9]. S. Agarwal, B. Mozafari, A. Panda, H. Milner,S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys '13, pages 29{42, New York, NY, USA, 2013. ACM.

[10]. M. O. Akinde, M. H. Bohlen,• T. Johnson, L. V. S. Lakshmanan, and D. Srivastava. E cient olap query processing in distributed data warehouses. In Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '02, pages 336{353, London, UK, UK, 2002. Springer-Verlag.

[11]. G. Alonso, B. Reinwald, and C. Mohan. Distributed data management in work ow environments. In RIDE, 1997.

[12]. A. Auradkar, C. Botev, S. Das, D. De Maagd, A. Feinberg, P. Ganti, L. Gao, B. Ghosh, K. Gopalakrishna, and B. Harris. Data infrastructure at linkedin. In 2012 IEEE 28th International Conference on Data Engineering, pages 1370{1381. IEEE, 2012.

[13]. The Big Data Benchmark. https://amplab.cs.berkeley.edu/benchmark/.

[14]. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. SIGMOD Rec., 1997.

[15]. A. Chervenak, E. Deelman, M. Livny, M.-H. Su,R. Schuler, S. Bharathi, G. Mehta, and K. Vahi. Data placement for scienti c applications in distributed environments. In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, GRID '07, pages 267{274, Washington, DC, USA, 2007. IEEE Computer Society.

[16]. R. Cole, F. Funke, L. Giakoumakis, W. Guy,A. Kemper, S. Krompass, H. Kuno, R. Nambiar,T. Neumann, M. Poess, K.-U. Sattler, M. Seibold, E. Simon, and F. Waas. The mixed workload CH-benCHmark. In DBTest '11.

[17]. B. F. Cooper, R. Ramakrishnan, U. Srivastava,A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. Proc. VLDB Endow., 1(2):1277{1288, Aug. 2008.

[18]. J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost,J. Furman, S. Ghemawat, A. Gubarev, C. Heiser,P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle,S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally-distributed database. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), pages 261{264, Hollywood, CA, Oct. 2012. USENIX Association.

[19]. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil,C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman,J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping

complex scienti c work ows onto distributed systems. Sci. Program., 13(3):219{237, July 2005.

[20]. A. Dey. Yahoo cross data-center data movement. http://yhoo.it/1nPRImNl, 2010.

[21]. A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess,A. Crolotte, and H.-A. Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, pages 1197{1208, New York, NY, USA,2013. ACM.

[22]. G. Giannikis, D. Makreshanski, G. Alonso, and D. Kossmann. Shared workload optimization. Proceedings of the VLDB Endowment, 7(6), 2014.

[23]. A. Gupta, F. Yang, J. Govig, A. Kirsch, K. Chan,K. Lai, S. Wu, S. Dhoot, A. Kumar, A. Agiwal, S. Bhansali, M. Hong, J. Cameron, M. Siddiqi,D. Jones, J. Shute, A. Gubarev, S. Venkataraman, and D.Agrawal. Mesa: Geo-replicated, near real-time, scalable data warehousing. In VLDB, 2014.

[24]. H. Herodotou and S. Babu. Pro ling, what-if analysis, and cost-based optimization of mapreduce programs. PVLDB, 2011.

[25]. J.-H. Hwang, Y. Xing, U. Cetintemel, and S. Zdonik. A cooperative, self-con guring high-availability solution for stream processing. In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, pages 176{185, April 2007.

[26]. K. Karanasos, A. Balmin, M. Kutsch, F. Ozcan,V. Ercegovac, C. Xia, and J. Jackson. Dynamically optimizing queries over large scale data platforms. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, pages 943{954, New York, NY, USA, 2014. ACM.

[27]. D. Kossmann. The state of the art in distributed query processing. ACM Comput. Surv., 2000.

[28]. T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete. MDCC: Multi-data center consistency. In EuroSys, 2013.

[29]. G. Lee, J. Lin, C. Liu, A. Lorek, and D. Ryaboy. The uni ed logging infrastructure for data analytics at Twitter. PVLDB, 2012.

[30]. H. Lim, H. Herodotou, and S. Babu. Stubby: A transformation-based optimizer for mapreduce work ows. Proc. VLDB Endow., 5(11):1196{1207, July 2012.

[31]. L. F. Mackert and G. M. Lohman. R optimizer validation and performance evaluation for distributed queries. In VLDB, 1986.

[32]. S. Madden. Database abstractions for managing sensor network data. Proc. of the IEEE, 98(11):1879{1886, 2010.

[33]. F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In SIGMOD, 2009.

[34]. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: A not-so-foreign language for data processing. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pages 1099{1110, New York, NY, USA, 2008. ACM.

[35]. A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pages 275{288, Seattle, WA, Apr. 2014. USENIX Association.

[36]. L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajan,I. Obermiller, and S. Shayandeh. Appinsight: Mobile app performance monitoring in the wild. In Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, OSDI'12, pages 107{120, Berkeley, CA, USA, 2012. USENIX Association.

[37]. M. Stonebraker, P. Brown, D. Zhang, and J. Becla. Scidb: A database management system

for applications with complex analytics. Computing in Science Engineering, 15(3):54{62, May 2013.

[38]. A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu. Data warehousing and analytics infrastructure at facebook. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pages 1013{1020, New York, NY, USA, 2010. ACM.

[39]. R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin,S. Shenker, and I. Stoica. Shark: Sql and rich analytics at scale. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, pages 13{24, New York, NY, USA, 2013. ACM.

[40]. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pages 2{2. USENIX Association, 2012.