D-WCPS: A Framework for Service Based Distributed Processing of Coverages

Michael Owonibi, Peter Baumann
Center for Advanced Systems Engineering (CASE)
Jacobs University
Bremen, Germany
{m.owonibi,p.baumann}@jacobs-university.de

***Abstract*— Distributed, service-oriented systems is often used today for geospatial data access and processing. However, it is difficult to find methods for easy, flexible, and automatic composition and orchestration of workflow of geo-services. More promising is the Open Geospatial Consortium (OGC) Web Coverage Processing Service (WCPS), which offers a multidimensional raster processing query language with formal semantics; we believe that this language contains sufficient information for an automatic orchestration. Based on this, we present the D-WCPS (Distributed WCPS) – a framework in which coverage processing workflow can be dynamically distributed among several WCPS servers. Every server can schedule and execute a query using information from its local WCPS registry. Each local registry, in turn, is mirrored across all servers. Some other contributions of this paper include query optimization algorithms, tuple-based parallelism, registry synchronization techniques. Several servers can, therefore, efficiently share data and computation with respect to dynamic, resource-aware coverages processing.**

Keywords - geoprocessing, query processing, distributed query processing, service registry, query optimization

I. INTRODUCTION

Current trend specifies the use of distributed, service-oriented systems for geospatial data access and processing. Some of the motivations for this include

- Availability high-speed networks.
- Computation intensiveness of either a single or workflow of geoprocessing tasks such that distributed processing pays off.
- Server limitations in terms of processing capability and applications installed.
- Distribution of dataset across several data centers.
- Increase in the geo-application requirements, which vary from simple 2-D and 3-D map visualization and download, to complex computation such as statistical analysis, data mining, image classification and ocean, atmosphere, and climate modeling.

Geo-services are usually, standardized by the Open Geospatial Consortium (OGC) and some of the standardized services include the OGC Web Coverage Service (WCS) for coverage data access [5]**;** the OGC Web Coverage Service-Transactional (WCS-T) used for adding, updating and deleting coverages on a server [4]**;** the OGC Web Processing Service (WPS) which defines a generic service that offers any sort of geoprocessing functionality over a network[25]**.** It turns out very difficult at least to find methods for a flexible, automatic orchestration of geo-services. Hence, geo-service orchestrations are typically, performed manually, such as in the case of cascading OGC WPS and Web Map Service (WMS) requests, and process-based compositions e.g., BPEL which hardwires the processing configuration. We claim that this is due to the lack of an explicit, machine-understandable semantics of these services. The Web Coverage Processing Service (WCPS) [23], however, accomplishes interoperability by defining a language for server-side processing of multi-dimensional spatial-temporal data. This language has a formal semantics definition, hence is machine readable and semantic web ready. To this end, this paper addresses the efficient answering of queries on large, complex spatial-temporal data sets distributed across a number of heterogeneous computing nodes. The aim is that incoming query requests, expressed in WCPS, are automatically split into sub-requests which then are processed by suitable nodes in the network to collectively produce the final result for the client. Task distribution can be based, among others, on the individual node capabilities and availability, and load situation, network capabilities, and source data location. Sample use cases for distributed processing include site suitability studies or statistical analysis using data available from different servers, and climate reconstruction using climate modelling algorithms.

The rest of the paper is structured as follows: Section II introduces the WCPS; related work is presented in Section III; we describe the distributed WCPS framework in Section IV, implementation and performance evaluation is presented in Section V and we conclude the paper in Section VI.

II. WEB COVERAGE PROCESSING SERVICE

WCPS specifies the syntax and semantics of a query language (service request) which allows for server-side retrieval and processing of multi-dimensional geospatial coverages representing sensor, image, or statistics data [23]**.** The term "coverage", in the general definition of OGC [20] and ISO [21], encompasses any spatial-temporally extended phenomenon. As currently overarching query languages in this generality are not sufficiently understood, WCPS focuses on raster data. The raster data is a gridded multi-dimensional array of some dimensionality, and some extent (spatial-temporal domain) where each grid cell value represent information. Sample raster data include 1-D sensor time series; 2-D satellite imagery; 3-D x/y/t image time

 ISBN: 978-1-61208-178-6

series, and 4-D atmospheric model data. WCPS queries are given as expressions composed from coverage-centric primitives. The grouping of these primitives is shown below:

- Geometric operations extract some subset of cells which together again form a coverage. Trimming retrieves a sub-coverage whose dimensionality remains unchanged. Slicing delivers a cut-out with reduced dimensionality.
- Induced operations apply cell type operations to all cells in coverage simultaneously. This includes arithmetic, trigonometric and logical operations, etc
- Coverage summarization includes aggregation operations like count, average, min, max etc.
- All of the above functions actually represent convenience operators which can be reduced to a coverage constructor, an aggregator, or a combination thereof.
- Scaling and reprojection constitute non-atomic function.
- Data format encoding specifies how results are to be shipped back to the client. The list of such encodings includes formats like TIFF, NetCDF, or SEG-Y.

Shaped in the style of XQuery and SQL, the WCPS defines a declarative, set-oriented query language for a coverage processing workflow. The overall structure of WCPS is as follows:

```
for     c1 in (C1,1, C1,2,…, C1,m),
        cn in (C2,1, C2,2,…, C2,m),
        …,
        cn in (Cn,1, Cn,2,…, Cn,m)
where   booleanExpr(c1, …, cn)
return  processingExpr(c1, …, cn)
```

This can be seen as a nested loop where each c_i is bound to the $C_{i,j}$ coverages in turn. For each variable binding, the "where" predicate *booleanExpr()* is evaluated first. Only if the boolean expression evaluates to true will the *processingExpr()* will be evaluated on the current variable assignment and its result element will be added to the result list. We introduce WCPS by way of example.

Assume a WCPS server offers 3-D satellite image time series stacks, S1 and S2, plus a 2-D bitmask M with the same spatial extent as the time series cubes. Then, the following query returns those cubes where, in time slice T, threshold value V is exceeded in the red band somewhere within the mask area:

```
for     s in ( S1, S2),
        m in ( M )
where   some(s.red[t(T)]> m and m>0)
return  encode( s/max(x), "netcdf" )
```

The subsetting operation in square brackets specifies a cut along axis *t*. The aggregator expression *some()* conflates this to a single Boolean value. Those result cubes which pass this filter are shipped to client in NetCDF format.

The WCPS query processing model is based on adapted rasdaman query processing model [27]. It consists of a set processing tree, and coverages processing trees as sub-trees

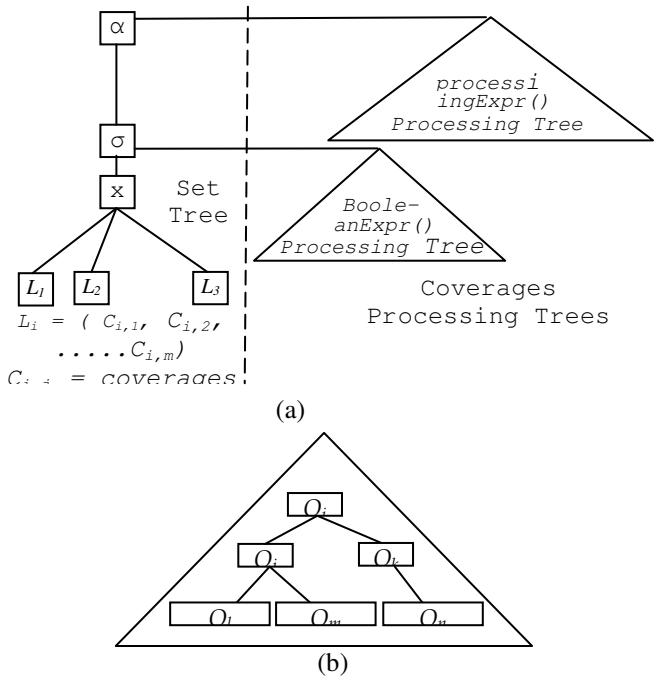

Figure 1: WCPS Query Tree

as shown in Figure 1(a). The set processing trees specifies the assignment of coverages to coverage iterator. The leafs of the set tree are the coverage lists. The set tree contain three relational operations – the cross (cartesian) product (*x*) of the different coverage lists, the selection (σ) of tuples from the resulting cross product table based on the predicate defined in the *"where"* clause, and the application (*α*) where the coverage processing expression is evaluated on the current iterators' coverages assignment. On the other hand, the coverage processing expressions, i.e., *booleanExpr()* and *processingExpr()*, are trees of coverage processing operators O_x as shown in the sample query tree in the Figure 1(b)

Due to the fact that the semantics of WCPS service request is known both to the client and servers, [22] opined that automatic service dispatching, chaining and optimizing is possible, as a WCPS server is able to automatically extract portions that are best resolved locally, distribute other parts to other suitable servers, re-collect results, package them, and return them without any human interference.

In this paper we described our distributed WCPS (D-WCPS) system wherein several servers can collaboratively and dynamically execute geo-processing tasks specified as WCPS query transparently. Servers can therefore share data, load and applications. We also describe the means of orchestrating the composed service efficiently in a fault-tolerant manner. Depending on the objective of the servers, different scheduling algorithms can be used for decomposing a query in such infrastructure. Hence, we do not specify the details of any particular scheduling algorithm in this paper.

III. RELATED WORKS

In classical distributed database systems (DBMS), grid computing systems, and Service Oriented Architecture

(SOA), a mediator-based method typically used for distributed query processing (DQP). In this approach [26], a mediator's registry stores and integrates the data sources schema, statistics and properties. It also contains the server properties of all the data servers**.** Global queries (queries which address more than one server) are directed to the mediator which parses and translates them into a query tree. Using information from the registry, the query tree is optimized logically and physically using both single and multi-node algorithms. The query tree operators are, later on, scheduled, and execution code is generated and run for the scheduled tree. Usually, the orchestration and integration of partial results from other servers are done in the mediator.

Some of the DBMS-based middleware using this approach include DISCO[3], Garlic [15], Hermes [28], TSIMMIS [12], Pegasus [18]. Also, several levels of support for a mediator-based data integration is provided by major database vendors such as IBM (using DB2 Propagator), Sybase (using Replication Server), Microsoft (SQL Server), and Oracle (using Data Access Gateways) [1][8]. Classical systems like R*[10] integrates data from several databases, while SDD-1[24], present mechanisms for distributed join processing algorithms on a homogenous set of servers.

Similarly, grid-based DQP typically consists of several mediators using one registry. Some of the generic grid-enabled query processors include Polar [13], OGSA-DQP[19], SkyQuery[30], CoDIMS-G [31]; and GridDB-Lite [29]. The main task of GridDB-Lite is the selection and transfer of distributed scientific data from storage cluster to compute clusters. Polar* is a distributed object oriented query processor which accesses remote data using remote operation calls. OGSA-DQP and CoDIMS-G are based on service oriented grid. OGSA-DQP extends the concepts in POLAR by automatically composing a static, per-client DQP service instance from a set of manually selected resources. CoDIMS-G, on the other hand, profiles services in order to select the resources to use, and adaptively reschedules query operators based on runtime conditions. SkyQuery provides an implementation of a mediator-based DQP for distributed astronomic data in a SOA. Its mediator dynamically tests for performance of the servers before scheduling its query.

However, with respect to distributed coverage processing, we note the following

- In DBMS, grid and SOA-based DQP, many of the scheduling algorithms, and parallelization, optimization, and execution model deals with relational and xml databases as opposed to coverages. For instance, a typical DBMS-based DQP will focus on scheduling relational join operators on servers based on different join algorithms. However, in D-WCPS, scheduling the costly coverage processing operators takes precedence over comparatively less expensive join operators.
- The centralized mediator used for the registry management and query execution constitutes a performance bottleneck and single point of failure.
- Concepts and framework of D-WCPS is based on SOA while DBMS-based DQP is typically not.
- Because the grid deals with heavy weight, long running scientific applications, the costs of the grid-based DQP scheduling and execution overhead is relatively small compared to the query execution cost. However, WCPS is typically, a medium weight application whose running time varies between milliseconds to minutes. Hence, cost of overhead of grid based methods in D-WCPS is significantly large and inefficient.

A component common to all distributed systems is the registry. These can go by different names such as Universal Description Discovery and Integration in SOA, OGC Catalog Service for the Web in OGC web services, federated database catalog in distributed databases, Monitoring & Discovery System in Globus Grid etc. To ease management overhead, registries are usually centrally located. However, besides constituting a performance bottleneck and single point of failure, this architecture is not usually efficient. Another proposal uses decentralized Peer to Peer registry based on distributed hash table [6]. Although this system scales up and is resilient to failure, they response times for a query is usually large. Also, its registry's management is complex, and it does not efficiently support range queries. Similarly, some other registries are based on meta-directory architecture whereby a node stores meta-information about the distributed registries [2]. This has the advantage of scaling, however, performance is still an issue and it is prone to single point of failure. Hence, we proposed the use of mirrored registry for D-WCPS. This has the advantage of being highly efficient in its query processing because it is available locally on every server. However, synchronization of all the servers for transaction based queries (updates, insert, delete) can be costly. As it is not expected that the rate of transaction in D-WCPS is going to be high, this cost will not be significant. Besides, since the database is mirrored on all servers, the registry is more resilient to failure, and the central server's performance bottleneck is removed.

Overall, our work focus on processing coverages and more emphasis is laid on optimizing, scheduling and executing coverage processing operators rather than relational operators. We also present an orchestration model which is based on recursive nesting of queries. Furthermore, we specify the use of inter-tuple parallelism in query execution. Lastly, we use an architecture where every participating server can serve as a mediator.

IV. D-WCPS FRAMEWORK

On a high level, we present D-WCPS - a framework where several WCPS servers can share the computation of a service request. Every server in this framework has a local copy of the global WCPS registry of data and services. Information from this registry is used to decompose a global query request into a distributed query request. As shown in Figure 2, every server runs WCPS and registry services, and these in turn, are made up of several components. The procedure of composition and execution of distributed WCPS is adapted from distributed DBMS-query processing. After the global query is received by any of the servers:

- The parser transforms the query to a query tree.

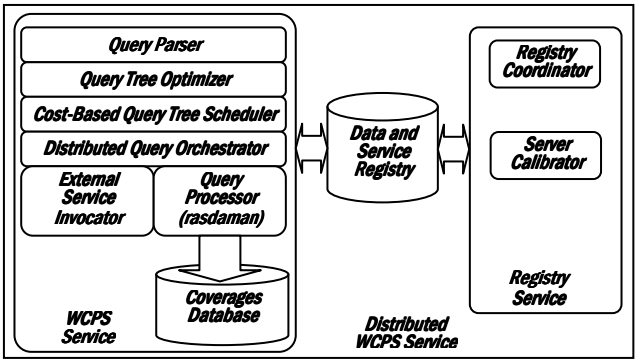

Figure 2: Components of a D-WCPS Server.

- The coverages metadata and location, and the servers to use for the distributed processing together with their capacities (e.g., CPU speed) and restrictions (e.g., maximum memory available) are then determined by querying the local registry.
- Based on the query structure, data sizes and location, the coverage processing query tree is optimized for distributed execution by rearranging its operators using different sets of equivalence rules.
- Using the server capabilities and initial data locations, the query is decomposed to a distributed query such that fulfils an objective function. Several objective functions exists and these include minimizing execution time, total data transferred, and total time spent on all servers; maximizing throughput; and load balancing on different servers. Decomposition of the query tree involves the scheduling of the query tree operators on different servers. This is an NP-complete problem, hence, the use of heuristic-based algorithms [26].
- After the scheduling, the global query is re-written into a distributed query (query with scheduling information) which is then executed.

A. *Inter-Tuple Parallelism in D-WCPS*

From Section II, the WCPS query tree consists of a set tree and coverage processing trees. The set tree creates a table of cartesian product of the coverage lists bounded to each coverage iterator in the query. Then, the predicate expression and/or processing expression are executed for each tuple in the table. Compared to the coverage processing operation, the cartesian product operation is cheap, hence, it is done on the server which receives the query. However, the execution of the coverage processing tree is distributed. Moreover, since each tuple in the cross product table can be processed independently of others, we parallelise the processing of each tuple in the table i.e., the optimization, scheduling and execution of a query are parallelized on a tuple-by-tuple basis. So, given the sample query below

```
For   a in (X, Y),
      b in ( Z )
where max(a) < avg(b)
return encode(cos(a)+min(b),"hdf")
```

Table 1: Cartesian Product Table of WCPS Query.

Tuples	Iterator a	Iterator b	Query Materialization
1	X	Z	Predicate : max(X) < avg (Z)
			Processing Expression: Encode(cos(X) + min(Z), "hdf")
2	Y		Predicate : max(Y) < avg(Z)
			Processing Expression: Encode (cos(Y) + min(Z), "hdf")

A cartesian product in Table 1 is created whose processing is then parallelized tuple-wise.

B. *Optimization*

In order to increase the efficiency of the execution of the D-WCPS, the global query is optimized by re-writing of the query based on equivalence rules. By query re-writing, we mean the re-arrangement of the ordering of operators of a query tree. This is done using a two-staged approach – applying single-node optimization before a multi-node optimization. We apply the optimizations on the coverage processing tree, hence, any reference to operator from henceforth implies coverage processing operator. The single node optimization assumes the query is executed on a server and is based on rasdaman query re-writing rules. The rationale and proof of the optimizations can be obtained from [7]. Overall, the idea is to minimize the size of the data processed by an operator. This is because, smaller input data for an operator implies

- Less processing work would be done by the operator
- Reduced data transfer time between an operator and its operand.

Some of the single node optimization includes pushing down of domain reducing, and aggregation expression down the query tree; using the associative and distributive properties of expression to re-write queries such that data transferred is minimized. For instance, assuming O_G represents coverage subsetting operation and O_C represents other coverage processing operations. By pushing down of the geometric operation as shown in Figure 3(a), the cost of executing O_C, and transferring data between O_C and O_G will be smaller. Similarly, in Fig 3(b), if operators *a, and b* are associative, their operands can be re-arranged such that data processed at and transferred from operator *b* is minimized.

The aim of multi-node optimization is to prepare a query

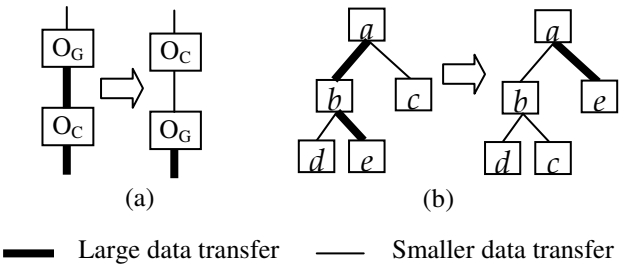

Figure 3: Single Node Query Optimization.

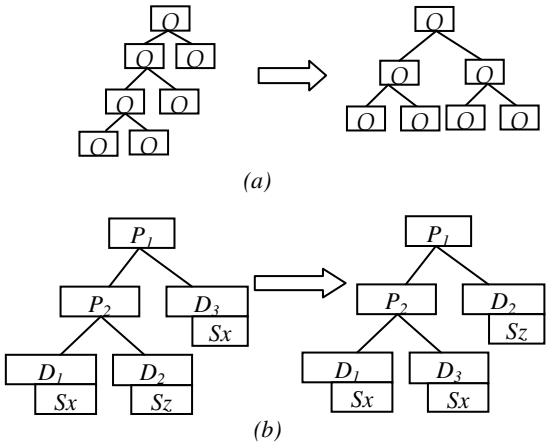

Figure 4: Multi-node Optimization.

tree for distributed execution. And one of the optimizations we introduce here aim at transforming a left deep tree to a bushy tree (Figure 4(a)) using the associative and distributed properties of the operators in the tree. This is because left deep tree would have a high through time; irrespective of scheduling algorithm used. Besides, more operators can be executed in parallel in bushy tree. Another optimization we used here is to use the bring operators which have data on the same host as close together as they can possibly get on the tree. This ensures that data will not be transferred just for integration purposes when there is no performance gain. As an example, consider the tree in Figure 4(b), where D_1, and D_3 are located on server Sx and D_2 on server Sz. If operators P_1 and P_2 are associative, operand D_2 will be interchanged with D_3 in order to ensure D_1 and D_2 are close to each other.

C. Scheduling

After the optimization of the query tree, the operators of the coverage processing trees are scheduled. Several algorithms exist for scheduling DAGS [32][9]. The choice of algorithm by a server is specified by used by the objective the server wants to fulfill. Different scheduling algorithms can therefore be used in D-WCPS. However, it has been shown that Heterogeneous Earliest Finish Time (HEFT) algorithm [11] is one of the best algorithms [9] in systems consisting of several heterogeneous servers.

D. Distributed Query Modeling and Its Orchestration

A P2P orchestration model, whereby WCPS servers recursively invoke other servers with a distributed query request, is used for executing D-WCPS. After a server receives a distributed query request, it invokes other servers with partial query requests as specified in the query, executes its local query requests and integrates the results. The integration may involve writing a temporary copy of the data due to the fact that partial query results can be larger than the main memory. The distributed query request is composed by the server which receives the initial global query. Therefore, other servers used for executing the distributed query need not run the scheduler again except if there is a change in the conditions in the network, and the query needs to be adapted.

```
For p in (
          For r in (
                    For t in (T)
                    return  encode(  cos(
                    t),"raw") on server_B
                   )
              s  in ( S )
              return encode((a+b),"hdf")
              on server_A
        )
    q  in ( Q )
    return encode  ( x + max(y), "tiff" )
```

Figure 5 : Nested Distributed WCPS Query.

The WCPS query syntax is modified to support such distributed execution [17]. In the introduced modifications, a coverage iterator will not only bind to coverages, but can as well bind to partial queries with a specified execution host. For example, Figure 5 shows a distributed WCPS query with different levels subquery nesting. The server that receives the query invokes *server_A* with its inner subquery, and *server_A* in turn invokes *server_B* with its inner subquery.

The quality of D-WCPS schedule generated initially can deteriorate if conditions in the network change during the execution its execution. Two major runtime disruption addressed in D-WCPS are server overload, and server unavailability. In the case that an overloaded server receives a partial query, the overloaded server reschedules the query for other servers. Similarly, if a server that is critical (it has some data or operations which is not available on other servers) to the query is unavailable, the query execution is terminated, otherwise, the partial query is rescheduled by the server which invokes the request on the unavailable server.

E. The Registry

The architecture of the registry adopted for a framework depends on the trade-offs between requirements such as efficiency, scalability, simplicity, availability, fault-tolerance, ease of management, flexibility, allowance for redundancy, rate of update, support for range or singleton query, ability to easily classify the registry entries. In D-WCPS, emphasis is placed on efficiency, fault-tolerance, and easy configuration. In addition, we envisage a system with a slow rate of update and which can scales to thousands in terms of services registered, and each server, in turn, can hold thousands of coverages. In this respect, we propose mirrored registry architecture. Because each server has a local copy of the registry, querying it for query decomposition information is very efficient compared to if the registry were to be external service. Furthermore, each local registry in the network is kept in sync with the others,

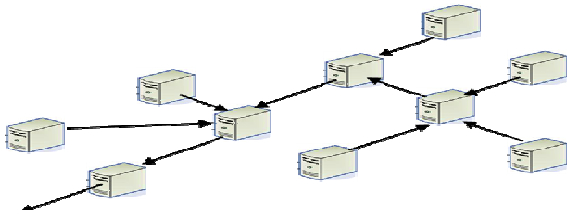

Figure 6: Mirrored Registry Synchronization.

 ISBN: 978-1-61208-178-6

and updates are only made when a new server joins the network, a new data is added to /deleted from a server, a server's properties change, or when the properties of some data saved is changed. Therefore, the challenge in the networks is keeping servers in sync with minimum effort.

The arrangement of the servers in the mirrored registry network is based a hierarchical topology for. Each server can only join the network from one server i.e., a server has only one gateway but it can have several backup gateways. And several servers can join the network through a server. In this paper, all the servers that share a gateway are child servers of their gateway server. A server can only receive a message from either its gateway or any of its child servers. When a server receives or generates a message, it propagates it to all the servers connected to it (child servers and gateway) except the source of the message. In this way, messages get sent to all the servers without looping as a broadcast will do. Furthermore, every server that intends to join the network can register with any server in the network. In order not to overload a server, a server has the maximum number of child servers it can have, and if any intending server wants to register with a server and its maximum number of child servers has been reached, it forwards the registration information to its child servers in a round-robin fashion. In case the gateway of a server becomes unavailable, the server can re-register with any other gateway server.

Three interfaces are exposed by the registry service namely register, update, and info interface. The registration interface is used for registering and de-registering servers. The update is used to insert, delete or update information about servers or data, and the info interface is used for management e.g., receiving and sending keep-alive messages, informing other peer servers that their gateway server is dead.

F. Calibrating and Profiling

Every server has a calibration engine which is used to measure its capacities for coverages processing. Information gathered by the engine are published into the service registry, and these include the read, write, and coverage processing speed and overhead; available memory; number of simultaneous processes that can run without degrading the servers performance; set of preferred servers (the set of servers a server will prefer to use in for distributed processing), and the network speed to these servers; and lastly, the average network speed to all servers. We obtain the read and write speed and overhead by writing some test data to and reading some test data from the database. The I/O speed and overhead depend on type of disk systems used (e.g., whether it is RAID system, virtual disk system etc), their speed and configurations, file/database system used and their configurations, etc. Similarly, [14][7] opines that the quoted speed of systems in terms of MIPS, FLOPS, QUIPS, clock rate cannot be used to determine the performance of a system due to factors such as inter-processor communication, I/O performance, cache coherence and speed, memory hierarchy etc. Therefore, we define the speed of processing of a system with regards to coverage processing as the speed it takes a system to do a copy of a char-typed pixel from one memory location to another. The calibration engine obtains this value by running a set of standardized query against the database. Furthermore, due to fact that systems nowadays have several physical and/or virtual processors, systems can run several queries simultaneously, without any significant degradation of its performance. Hence, we determine the maximum number of queries a system can run without performance loss.

V. IMPLEMENTATION AND EVALUATION

An experimental D-WCPS infrastructure was set up consisting of 30 WCPS servers which are heterogeneous with respect to processing, read, write and network speed. A server was chosen as the initial D-WCPS server with which other servers have to register before they can start registering other servers. For this setup, we choose minimization of execution duration as our objective. Therefore, a modified form of HEFT algorithm [11] based on a coverage processing cost model [16] was used to schedule the operators of any query received by any of the servers. Using query trees with different types of structural properties, operators, and initial data distribution, we evaluate the performance of our framework with respect to some of the query processing and optimization techniques.

In Figure 7(a), we highlight the gains of pushing down subsetting operators in a distributed processing system, given the percentage of the total initial data retrieved by the subsetting operator. Due to the large processing and inter-server data transfer costs, the smaller the percentage gets the smaller the execution duration. Similarly, Figure 7(b) compares the query execution times when a left deep tree is transformed to bushy tree and when it is not. In some of the cases, the execution duration does not change, but in many others, it is reduced. The performance gain of inter-tuple parallelism is also shown in Figure 7(c).

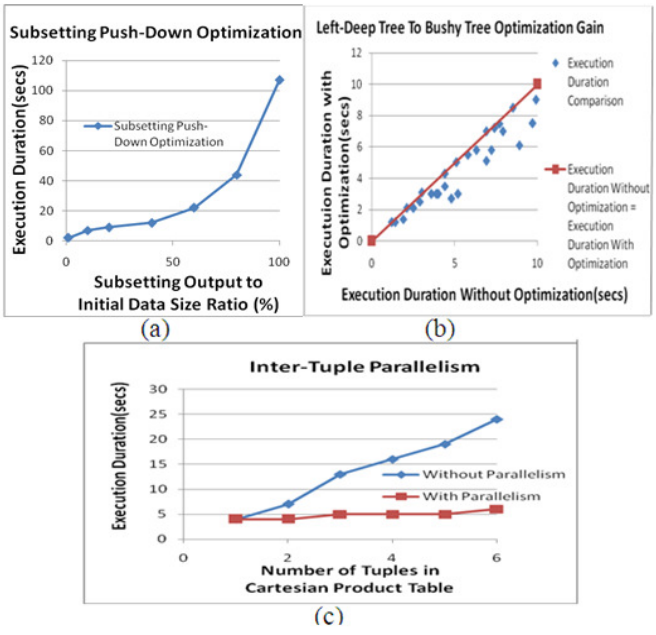

Figure 7: D-WCPS Performance Evaluation Graphs

VI. CONCLUSION

The OGC WCPS offers a multidimensional raster processing query language with a formal semantics which contain sufficient information for automatic orchestration. Based on this, we present an infrastructure for distributed execution of WCPS query by dynamically composing services from a query request. Every server in the network has a local copy of the WCPS registry, and servers synchronize the updates to the registry with each other. Using the information from the registry, and tuple-wise based parallelization, servers can optimize, decompose and execute a received query. Finally, we present the model for a decentralized orchestration of the distributed WCPS query. Several servers can, therefore, efficiently collaborate in sharing data, computation, loads and other tasks with respect to dynamic, resource-aware coverages processing.

ACKNOWLEDGMENT

Many thanks to the Earth System Science Research School (http://earth-system-science.org/), and the European Union funded "EarthServer: Ad-hoc Analytics on Big Data" project (http://earthserver.eu/) for supporting this research.

REFERENCES

[1] A. Gounaris(2005). Resource aware query processing on the grid. Thesis report, University of Manchester, Faculty of Engineering and Physical Sciences.

[2] A. Kassim, B. Esfandiari, S. Majumdar, and L. Serghi, A flexible hybrid architecture for management of distributed web service registries, in Communication Networks and Services Research (CNSR), vol. 5, 2007

[3] A. Tomasic, L. Rashid, and P. Valduriez, Scaling heterogeneous database and design of DISCO, in Proceedings of the 16th International Conference on Distributing Computing Systems (ICDCS), Hong Kong, May 1996.

[4] A. Whiteside (ed), Web Coverage Service (WCS) —Transaction operation extension OGC 07-068r4 Version: 1.1.4 , 2009.

[5] A. Whiteside, and Evans J.D. 2008. Web Coverage Service (WCS) Implementation Standard, 07-067r5.

[6] C. Schmidt, and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," World Wide Web Journal, Vol. 7, No. 2, June 2004, pp. 211-229.

[7] D. J.Lilja. 2000. Measuring computer performance: a practitioner's guide, Cambridge University Press, New York.

[8] D. Kossmann. 2000. The State of the Art in Distributed Query Processing. ACM Comput. Surv., 32(4):422–469, December 2000.

[9] F. Dong, and S.K. Akl, Scheduling algorithms for grid computing: State of the art and open problems, Technical Report No. 2006-504, School of Computing, Queen's University, Kingston, Ontario, Canada, January 2006.

[10] F. Mackert, and M. Lohman, R* Optimizer Validation and Performance Evaluation for Distributed Queries, Proceedings of the 12th International Conference on Very Large Data Bases, p.149-159, August 25-28, 1986.

[11] H Topcuoglu, S Hariri and M.-Y Wu, Task scheduling algorithms for heterogeneous processors, 8th IEEE Heterogeneous Computing Workshop (HCW '99) (1999), p. 3–14.

[12] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, V. Vassalos, and J. Widom, The TSIMMIS approach to mediation: Data models and languages, Journal of Intelligent Information Systems, vol. 8, no. 2, 1997.

[13] J. Smith, A. Gounaris, P. Watson, N. Paton, A Fernandes, and R. Sakellariou. Distributed query processing on the grid. International Journal of High Performance Computing Applications, 17(4), 2003.

[14] J.L. Gustafson, and Q. O. Snell, HINT: A New Way To Measure Computer Performance, Proceedings of the Twenty- Eighth Hawaii International Conference on System Sciences HICSS-95.

[15] M. J. Carey , L. M. Haas , P. M. Schwarz , M. Arya , W. F. Cody , R. Fagin , M. Flickner , A. W. Luniewski , W. Niblack , D. Petkovic , J. Thomas , J. H. Williams , and E. L. Wimmers, Towards heterogeneous multimedia information systems: the Garlic approach, Proceedings of the 5th International Workshop on Research Issues in Data Engineering-Distributed Object Management (RIDE-DOM'95), p.124, March 06-07, 1995.

[16] M. Owonibi and P. Baumann, A cost model for distributed coverage processing services. In Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems 2010. ACM, New York, NY, USA.

[17] M. Owonibi, and P. Baumann, 2010: Heuristic Geo Query Decomposition and Orchestration in a SOA. InProceedings of the 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS '10).

[18] M.C. Shan, Pegasus architecture and design principles, in Proceedings of the ACMSIGMOD International Conference on Management of Data, Washington, DC, USA, June 1993.

[19] M.N. Alpdemir, A. Mukherjee, A. Gounaris, N.W. Paton, P. Watson, and A.A.A Fernandes, OGSA-DQP: A Grid Service for Distributed Querying on the Grid. EDBT 2004. LNCS, vol. 2992, pp. 858–861. Springer, Heidelberg (2004).

[20] N.n. 2007 Abstract Specification Topic 6: Schema for coverage geometry and functions. OGC 07-011.

[21] N.n., 2008 Geographic Information - Coverage Geometry and Functions. ISO 19123:2005.

[22] P. Baumann and S. Keens, (2007). OWS-4 Workflow IPR Workflow descriptions and lessons learned OGC 06-187r1 version0.0.9. OGC Discussion Paper, OGC.

[23] P. Baumann, (ed.),2008. Web Coverage Processing Service (WCPS) Language Interface Standard. OGC 08-068r2.

[24] P. Bernstein , N. Goodman , E. Wong , C Reeve , and J Rothnie, Jr., Query processing in a system for distributed databases (SDD-1), ACM Transactions on Database Systems (TODS), v.6 n.4, p.602-625, Dec. 1981 [doi>10.1145/319628.319650].

[25] P. Schut (ed.), 2007. OpenGIS Web Processing Service. OGC 05-007r7 version 1.0.0.

[26] R. Ramakrishnan, and J. Gehrke. 2007. Database Management Systems (Third Edition), McGraw Hill, 2007.

[27] R. Ritsch, "Optimization and Evaluation of Array Queries in Database Management Systems". PhD Thesis, TU Muenchen, 1999.

[28] S. Adali, K.S. Candan, Y. Papakonstantinou, and V.S. Subrahmanian, "Query caching and optimization in distributed mediator systems," in Proceeedings of ACM SIGMOD International Conference on Management of Data, Montreal, Canada, June 1996.

[29] S. Narayanan, T. Kurc, U. Catalyurek, and J. Saltz, Database support for data-driven scientific applications in the grid. Parallel Processing Letters. v13 i2. 245-271.

[30] T. Malik, A Szalay, T. Budavari, and A. Thakar. SkyQuery: A web service approach to federate databases. In CIDR,2003.

[31] V. Fontes , B. Schulze , M. Dutra , F. Porto , and A. Barbosa, CoDIMS-G: a data and program integration service for the grid, Proceedings of the 2nd workshop on Middleware for grid computing, p.29-34, October 18-22, 2004, Toronto, Ontario, Canada [doi>10.1145/1028493.1028498].

[32] Y. Kwok and I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Computing Surveys (CSUR), v.31 n.4, p.406-471, Dec. 1999 [doi>10.1145/344588.344618]

 ISBN: 978-1-61208-178-6