Strategies to implement Edge Computing in a P2P Pervasive Grid

Luiz Angelo Steffenel, Manuele Kirsch Pinheiro, Lucas Vaz Peres, Damaris Kirsch Pinheiro

Strategies to Implement Edge Computing in a P2P Pervasive Grid

Luiz Angelo Steffenel, University of Reims Champagne-Ardenne, Reims, France
Manuele Kirsch Pinheiro, Pantheon-Sorbonne University, Paris, France
Lucas Vaz Peres, Federal University of Western Pará, Santarém, PA, Brazil
Damaris Kirsch Pinheiro, Federal University of Santa Maria, Santa Maria, Brazil

ABSTRACT

The exponential dissemination of proximity computing devices (smartphones, tablets, nanocomputers, etc.) raises important questions on how to transmit, store and analyze data in networks integrating those devices. New approaches like edge computing aim at delegating part of the work to devices in the "edge" of the network. In this article, the focus is on the use of pervasive grids to implement edge computing and leverage such challenges, especially the strategies to ensure data proximity and context awareness, two factors that impact the performance of big data analyses in distributed systems. This article discusses the limitations of traditional big data computing platforms and introduces the principles and challenges to implement edge computing over pervasive grids. Finally, using CloudFIT, a distributed computing platform, the authors illustrate the deployment of a real geophysical application on a pervasive network.

KEYWORDS

Big Data, Computing Middleware, Distributed Computing, Edge Computing, Ozone, P2P, Pervasive Grids

INTRODUCTION

Big data and data analytics have become essential tools for the strategic planning of any company. While data analysis is not a new topic, it was boosted by the development of large-scale computing platforms, notably the clouds. While cloud computing relies on distant resources, several works try to leverage the use of proximity resources as effective computing platforms (Garcia Lopez, 2015; Parashar & Pierson, 2010; Steffenel & Kirsch-Pinheiro, 2015).

Indeed, the number and nature of proximity computing devices (smartphones, Internet of Things - IoT, etc.) is growing exponentially, and it is important to understand how to exploit the power of these computing resources. For this reason, new approaches like edge or fog computing aim at delegating part of the work to devices in the "edge" of the network (Lopez, 2015). Because several strategies can be used to implement edge computing, this work specifically focus on the use of pervasive grids to leverage such challenges. Indeed, pervasive grids (Parashar & Pierson, 2010) associate classical and volatile computing resources. We believe that organizations can perform big data analytics with minimal costs by associating IoT and mobile devices as well as idle or unused resources in the enterprise network.

Therefore, in this paper, we discuss the limitations of traditional big data computing platforms and introduce the principles and challenges to implement edge computing over pervasive grids. To

illustrate this, we present CloudFIT, a distributed computing platform based on a P2P overlay, and discuss how it can be improved to efficiently deploy edge/pervasive computing applications, especially those related to big data analytics. Indeed, after presenting the main architecture of CloudFIT, we focus on the required strategies to ensure data proximity and context awareness, two factors that impact the performance of big data analysis in distributed systems.

For conducting this research, we adopted a two-fold approach, combining a conceptual research method with a case study a case study. Indeed, according to research method categories pointed out by Mora et al. (2008), a conceptual research corresponds to the study of ideas related to real objects including designing of new conceptual artifacts such as a framework/model, a method/model, or a system/component. For these authors, a "conceptual design research is the purposeful design of conceptual artifacts", in which the design artifact is dictated by the design goals. The principles and challenges we discuss in this paper represent, in our research, these design goals that guided the application of CloudFIT platform. The results of this conceptual research are then confronted to a case study issue from a real geophysical problem. Peres, 2013 conduct a case study analysis of the detection of Ozone Secondary Events (OSE) problem and Peres et al., (2017) present a detailed description of TOC monitoring by Brewer spectrophotometer in Southern Space Observatory SSO/CRS/INPE – MCTI (29.4 °S; 53.8°O; 488.7m) station for more than twenty years (1992 - 2014). Through this two-fold approach, we search for confronting our design goals with results from an empirical research proposed by the case study. Thus, we deploy the OSE detection algorithm over different scenarios representing edge and pervasive computing networks, both to validate the algorithm and to infer its execution performance.

This remain of this paper is organized as follows: we start presenting big data, the limitations of traditional computing platforms and the notions of edge computing and pervasive grids. The next section introduces the distributed computing platform CloudFIT and explain its main features. This section is followed by a case study that illustrates the usage of CloudFIT with a real application. Finally, we conclude this paper and explore future research directions.

BACKGROUND

Handling Big Data in Traditional and Pervasive Environments

Big data is research area presenting several definitions as it can be used in countless domains (Babiceanu & Seker, 2016). Even though, the literature often characterizes big data through a set of dimensions (Jagadish et al., 2014; Hashem et al., 2015; Gartner 2011; Babiceanu & Seker, 2016): *Volume, Variety, Velocity, Veracity* and *Value*. These 5 V's push big data to much more than a simple volume threshold as pointed out by Hashem et al. (2015), who stands that big data refers to "the increase in the volume of data that are difficult to store, process, and analyze through traditional database technologies". Indeed, the main challenges in big data processing are related to the complexity of the data itself, as for example one Terabyte of satellite images cannot be explored in the same way than 1 TB of structured text data.

As all five V dimensions have an impact on the way information should be handled, it is important to understand how traditional and pervasive computing platforms can be employed and what are their limitations.

Constraints and limitations of traditional infrastructures for big dataTraditional big data tools like Apache Hadoop (2016) were particularly designed for dedicated infrastructures like clusters or datacenters, relying on fast network connections and high-performance hardware, as underlined by Wright (2014). However, the usage of such infrastructures represent a significant investment, both on the acquisition of the dedicated equipment and its maintenance cost (human and material).

Public cloud platforms are often presented as a low-cost and scalable alternative to cluster infrastructures, thanks to their on-demand model. In a cloud platform, there is no need for investment

on material or maintenance, as these costs are assumed by the cloud provider. Although less expensive than cluster infrastructures, public cloud platforms have some drawbacks that must be evaluated when deploying big data applications.

One of the limitations of public cloud platforms we can cite is the transfer cost of a large volume of data through the network. Not only most cloud providers charge for large inbound/outbound traffic, but the connection speed may be a barrier on zones disposing of a poor or limited network access. In addition, transferring data to an external public cloud platform can represent a confidentiality risk that may prevent some applications to rely on public cloud platforms.

Because of these issues, some organizations may refrain to adopt high performance infrastructures like clusters and public cloud platforms for their applications. In the next section, we present a possible alternative platform for big data analysis for such organizations.

Proximity Services with Edge Computing and Pervasive Grids

The dissemination of proximity devices with non-negligible processing capacities (smartphones, tablets, laptops and nanocomputers like the Raspberry Pi) encourages the integration of these devices in the computing effort. Today, several works try to leverage the use of these proximity resources, and we strongly believe this can be achieved through the use of edge computing over pervasive grids.

Edge computing is a concept that aims at migrating part of the computation to devices in the "edge" of the network (Garcia Lopez et al., 2015). The main reason for this migration is the recent rise in computing power from mobile and proximity devices, transforming close base stations into "intelligent service hubs that are capable of delivering highly personalized services directly from the very edge" (Vermesan et al., 2014). Similar concepts like mobile edge computing (Dey et al., 2013; ETSI, 2014), edge-centric computing (Garcia Lopez et al., 2015) or fog computing (CISCO, 2013; Bonomi et al., 2012) also try to deploy applications and services closer to the final user, and can therefore be considered as variants of the edge computing concept.

Among the typical examples of edge computing we can cite fog services (Bonomi et al., 2012) and cloudlets (Satyanarayanan et al., 2009), all proposing the deployment of proximity servers offering enough computing power to perform complex computations (services) with a reduced service latency. In most cases, tasks are migrated from the cloud to the edge thanks to containers and microservice components (Pahl & Lee, 2015), but this is not a rule. In the same way, the computer power that can be offered by IoT devices, nanocomputers or even tablets and smartphones is often underestimated. With a few exceptions like (Dey et al., 2013), these works limit the role of edge devices by considering them as a frontend layer, connected to a bigger and more powerful "core" network that performs most of the work.

In our understanding, the notion of pervasive grid can be employed to unleash the latent computing power of proximity devices. Indeed, the concept of Pervasive grids (Parashar & Pierson, 2010) aims at transparently integrate sensing/actuating instruments and devices together with classical high-performance systems in a dynamic network. These grids can be composed by idle and under-explored resources in the enterprise network, by small Raspberry Pi or TV set-top devices, but also interconnect them to virtual machines deployed on cluster infrastructures. More than all, pervasive grids represent an opportunity to deploy computing tasks over local computing resources, minimizing data transfer over distant network.

Due to the heterogeneous nature of devices in a pervasive grid, tasks must be assigned according to the capabilities of each device, exploring the diversity of resources and improving the usage of proximity nodes. One example of such usage is presented by (Ramakrishnan et al., 2014), in which the computing resources of a home (laptops, tablets or nanocomputers) are associated to perform a preliminary analysis on sensor data concerning the movement of the residents, triggering an alarm or calling for an external action if necessary.

Of course, adopting pervasive grids for big data analysis implies considering several questions such as data processing, data distribution and tasks scheduling, all while efficiently matching the

resources capabilities (Shekar & Gokhale, 2017). In order to illustrate these challenges, we present in the next section a platform for pervasive grids, and discuss how it can be improved to deploy big data applications under the edge computing approach.

DEVELOPING A PLATFORM FOR EDGE AND PERVASIVE COMPUTING

The distributed computing concept accelerates in the 90's, as an extension on the Internet of the cycle stealing principle. Former applications aimed to crack RC5 or DES keys by exhaustive search thanks to the aggregation of hundreds of PCs to solve a problem. Web-based Computing projects arose at the end of the 90's like SETI@home (Anderson et al., 2002) and Boinc (Anderson, 2005), and were soon followed by P2P-based middleware (Brasileiro et al., 2007). However, the fast spread of clusters, grids and cloud infrastructures stalled the development of such middlewares, which were relegated to niche domains like those related to computing-intensive applications (combinatorial research, cryptography, etc.).

In this section, we present CloudFIT, a distributing computing middleware adapted to both computing and data-intensive applications. CloudFIT is structured around collaborative nodes connected over a P2P overlay network that provides communication, fault-tolerance and distributed storage, while its scheduling mechanism is based on the FIIT (Finite Independent Irregular Tasks) paradigm. For instance, applications that can be parallelized in a finite number of tasks and executed in batches are fit for this platform, like for example combinatorial problems (Krajecki & Jaillet, 2004) or ETL (Extract-Transform-Load) steps in a big data application. Hence, the well-known map-reduce paradigm used in several big data applications can be considered as a subset of the FIIT problems.

Instead of relying on containers and microservices, CloudFIT is written in Java and packed in a small jar file so that it can be easily deployed over a wide range of devices, from dedicated servers to low-end devices like Raspberry Pi. Because CloudFIT relies on P2P overlays, it is extremely elastic as nodes can join or leave the platform according to the availability of the resources or the variation in the demand. Also, inner services for replication and decentralized management of tasks ensure the completion of the tasks even in situations of high volatility or network partition (see Figure 1).

An application running on CloudFIT must simply implements two methods: how many tasks to solve and how to compute an individual task. These methods guide the deployment of tasks and their execution. Furthermore, each node receives the parameters of the current job and is able to locally decide which tasks still need to be computed and how to proceed, carrying the work autonomously if no other node can be contacted. Access to a distributed storage facility is also provided by the P2P overlay, allowing nodes to obtain input data and store their partial results. The status of completed tasks is distributed among the nodes, contributing therefore to the coordination of the work and to form a global view of the execution.

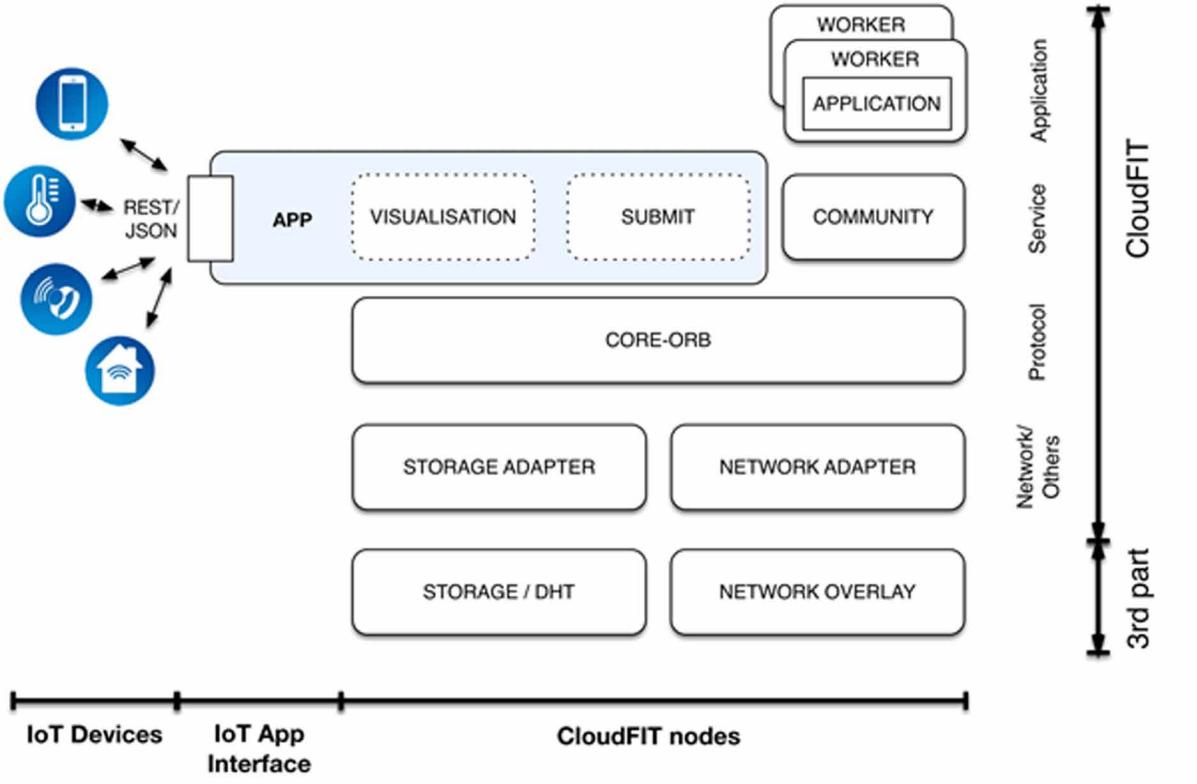
As nodes in a pervasive cluster may leave the network for different reasons (failure, low battery, network disconnection, etc.), CloudFIT supports by default a robust distributed scheduling with no "task reservation", allowing nodes to execute different tasks in parallel when they are able to communicate to each other or, in the worst case, to complete all the computation by itself. This scheduler mechanism also allows idle processes to speculatively execute incomplete tasks, reducing the "tail effect" when a task is computed by a slow node. The scheduling mechanism supports task and job dependencies (allowing the composition of DAGs and workflows) and can be also be driven by a context module (Cassales et al., 2016) with additional information about the nodes capacities.

The next sections give some details on the implementation strategies employed in CloudFIT.

Coordination and Clustering

One of the major challenges with edge computing is to coordinate which tasks are assigned to each resource in order to efficiently perform operations and reduce the communication latency (Shekar & Gokhale, 2017). When using a P2P overlay, however, we observe that nodes are often organized

Figure 1. CloudFIT architecture stack



indistinctly from their real location, preventing therefore a good use of close-range devices to provide a low latency service. We consider that P2P overlays must be enriched through the use of clustering, organizing nodes in computing layers that provide bounded communication latencies, context-awareness or trustworthiness/isolation.

Several clustering approaches are proposed in the literature (Johnen & Mekhaldi, 2011), with both manual or automatic clustering depending on specific metrics, so in CloudFIT we decided to implement clustering through the concept of community. As presented by Lim and Conan (2014), a community is a group ID to which nodes subscribe in order to share tasks and data or interconnect different communities in a multi-layered architecture. For instance, all applications in CloudFIT share a baseline community used for task distribution and wide-range communication, while the creation or subscription to additional communities can be managed directly by the applications.

Data Access

Another important aspect to consider of is how data is accessed, as big data applications involve the gathering, the transformation and the analysis of data. While these applications can rely on an external storage servers/services (like a cloud storage service), this solution is not always adapted to their needs as it incurs extra latencies or transfer fees.

In a previous work (Steffenel & Kirsch-Pinheiro, 2015), we conducted performance tests comparing the performance of Hadoop and CloudFIT when running the well-known *WordCount* application in a cluster platform. While these tests show that CloudFIT can reach performances at the same level than Apache Hadoop, we also observed the need to reinforce the *data locality,* i.e., the optimal data access for the computing tasks. Indeed, part of the success of Apache Hadoop is its capability to start tasks where the data is stored, avoiding therefore unnecessary network transfers.

Unfortunately, traditional P2P systems favor data spread and replication (to prevent the loss of data in the case of churn) at the expense of losing data locality (Wu et al., 2005). Indeed, P2P

storage APIs are often based on distribute hash tables (DHT), which are conceived to spread and replicate the data across the network, sometimes storing data on nodes really far from the original source (or the clients). In a DHT, the data is identified by a *hash id* that maps to a *node id* the entire P2P network. While this node may hold the primary data replica, it can also be a simple directory service pointing to the node where the data is really stored, making hard the mapping between tasks and data (Wang et al., 2015).

While CloudFIT is still based on a P2P overlay that lacks data locality information, we designed a solution to reinforce through *data proximity*. Indeed, our approach helps preventing too much dispersion of the data over the network, keeping therefore data close to the nodes that will perform the tasks.

For such, we rely on the TomP2P overlay (Bocek, 2015), which contrarily to most P2P overlays offers several hash keys (instead of a single hash key). While a typical P2P storage uses a simple mapping where the data is indexed in the node with ID closes to the hash key of the data, TomP2P identifies resources with up to four keys $\{k_l, k_d, k_c, k_v\}$, namely k_l (*location key*, which determines the node ID closest to the hash key), k_d (domain key, used for namespacing), k_c (content key, which identifies different resources stored in the same location) and k_v (version key, which allows the managing of different versions of the same resource).

In order to implement data proximity, we manipulate the location key so that it is not randomly spread among all nodes but specifically attached to a set of nodes. Thanks to a double hashing function, we decouple the location key and the content key for a resource: in a first moment, the content key is obtained through a traditional hashing method. Later, the location key is computed to map only among the community nodes.

Figure 2 shows an example of such mapping that reinforces the data proximity. Hence, in a traditional P2P storage with a single location key, a resource r_3 could be stored in any node in the network, depending on the hash result (like for example $hash(r_3)=k_{l7}$). By using a location and a content key and a community-aware hash function $hash_{ca}()$, we can bound the location key to the nodes in the community, all while properly identifying a resource. Hence, for a resource r_3 and a community C_1 composed by nodes with IDs k_{l2}, k_{l3} and k_{l4}, we can compute a community-aware location key k_l' that points to a node from the community. Hence, the primary copy of the resource r_3 will be located in the node k_l' with the content key k_{c3}. Even if the community is small data is preserved as each resource has its own content key. The domain key and version key can also be used to improve this resiliency.

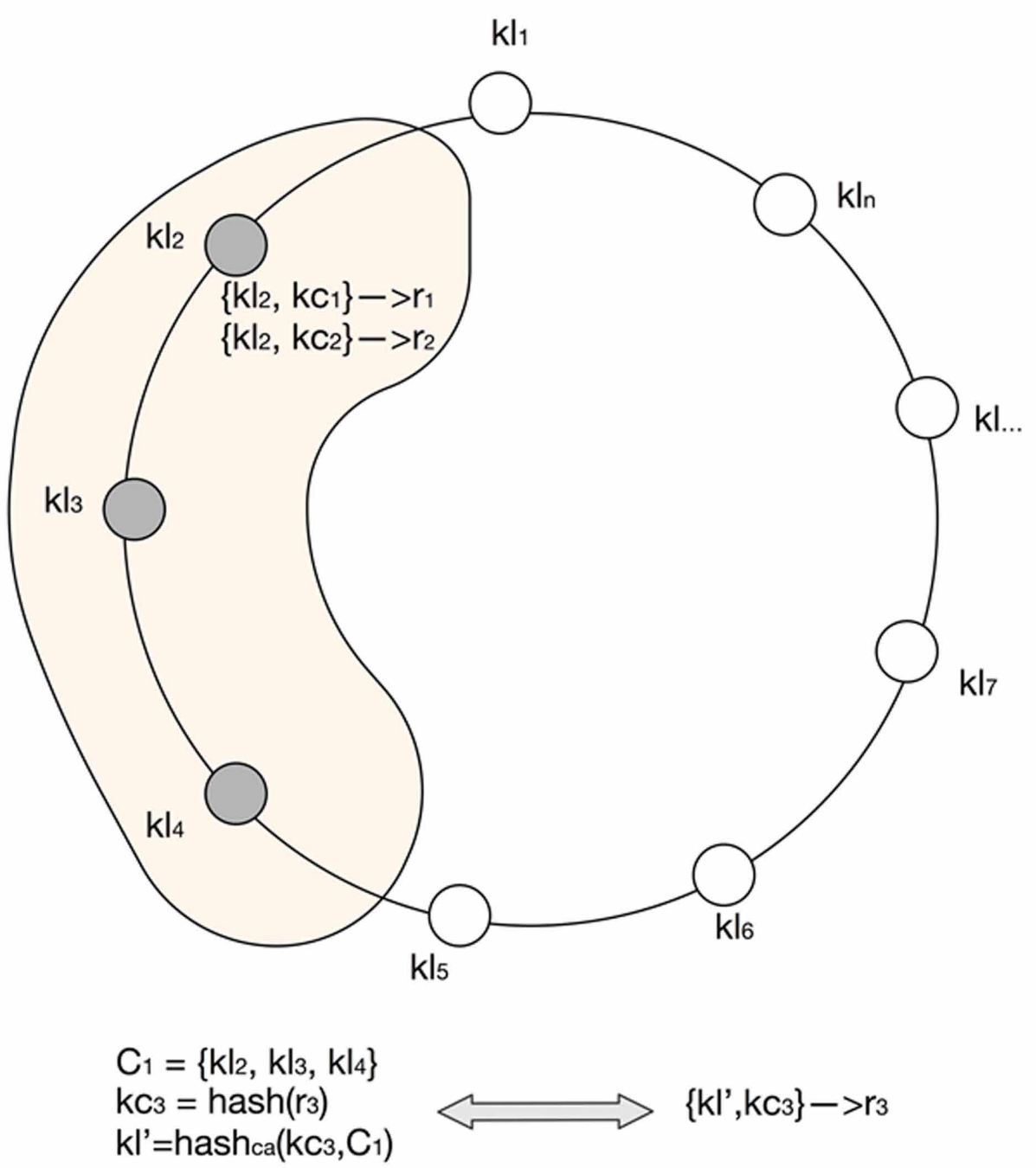
As a consequence, this mapping improves the probability that the primary copy of a resource is stored in a node from the community, all while allowing the storage overlay to perform replication on other nodes, even those outside the community.

Context and Scheduling

While the previous factors help improving the performance in a distributed computing platform, we shall address a last issue in order to offer a really scalable pervasive support. As the computing performance varies a lot from device to device, context information such as processing power, available memory and storage space or even current CPU load can be useful to improve the execution performance, as demonstrated by Dey et al. (2013). Also, efficiently matching the tasks with the resources capabilities and their locations is a key element on the optimization of performance-sensitive edge application (Shekar & Gokhale, 2017).

As presented before, the default scheduler on CloudFIT implements a best-effort algorithm that has the advantage of being totally distributed, i.e., all nodes collaborate to consume the tasks without a central coordinator. This scheduler currently performs a basic matching according to the tasks expressed requirements (minimum requested memory, disk space, etc.), but it can be enriched with additional context elements such as the CPU or network speeds (Celaya & Arronategui, 2011). This way, nodes can decide whether it is important to prioritize one single task at time in order to avoid memory swap (like in memory-intensive applications) or how to balance the available cores in the machine with the relative performance of its processors (a 4-core Raspberry Pi is still less powerful

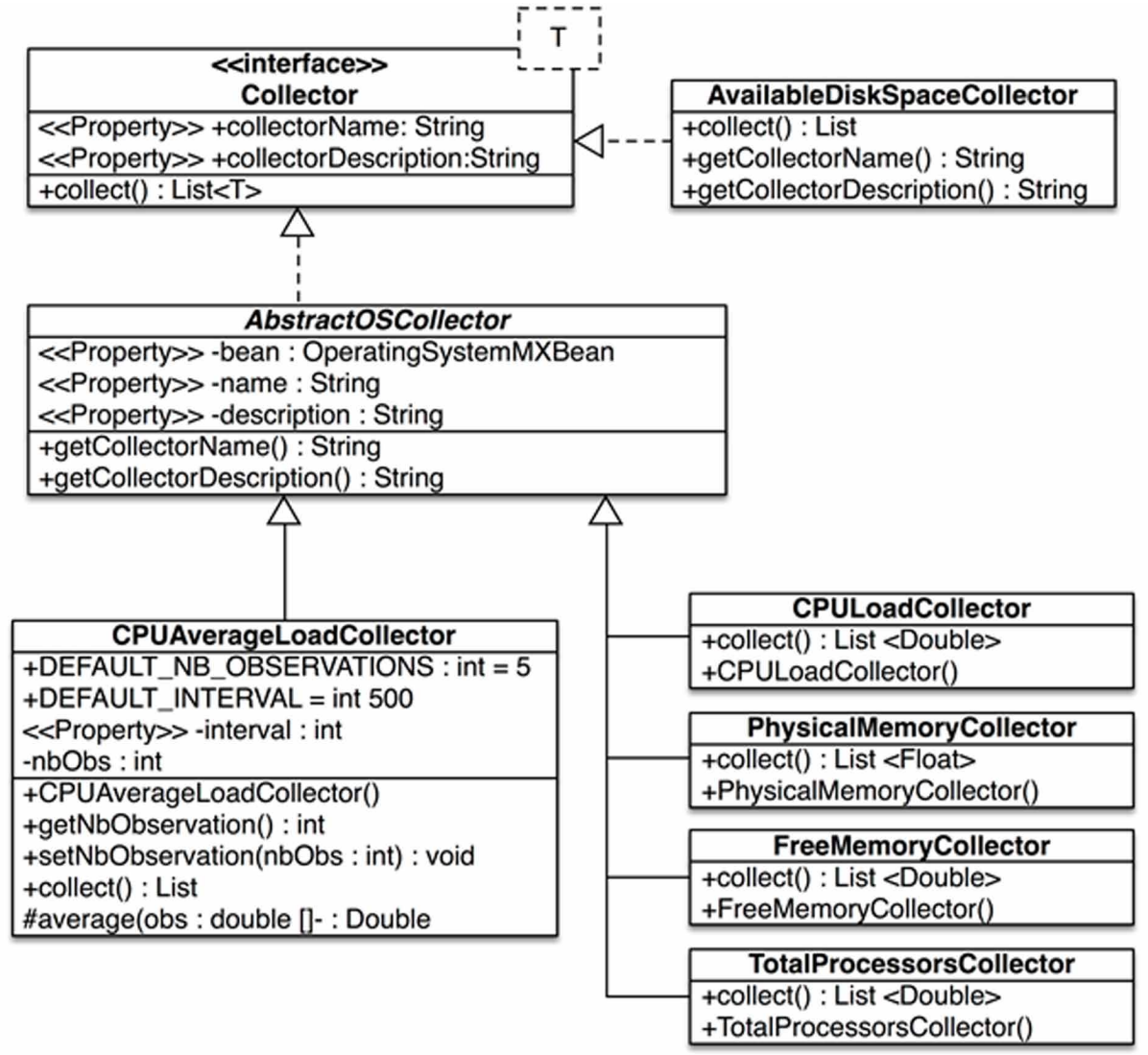
Figure 2. Computing the Location key to implement data-proximity



than a single core in an Intel i7 processor). Such context information is collected by a context collector as the one presented in Figure 3, which is integrated into the CloudFIT stack.

While the execution performance is a key component of context information, we also try to use it to improve the data access performance in the storage layer. Indeed, we observed in our experiments that data access performance is the result of both data access on the nodes (both on memory and disk) and the overhead caused by the storage management, affecting especially devices with low capacity. For example, a Raspberry Pi is highly penalized by the speed access of its SD card, in spite of having a good computing power. To circumvent such limitation, CloudFIT allows nodes to choose between acting as full storage nodes or simply as remote data clients. This way, low-end nodes can

Figure 3. Context collector structure (Cassales et al., 2016)



keep saving/reading data through the network but do not need to manage the storage, alleviating both the disk usage and the storage processing.

The next section presents a case study in which we design and deploy a data intensive application over a pervasive cluster using CloudFIT.

CASE STUDY: MONITORING OZONE EVENTS FOR UV ALERTS

In order to better validate design assumptions proposed on CloudFIT, we have decided to confront those with an empirical environment, through a case study issue from a reality situation. Thus, in this section, we present a real application developed over CloudFIT and deployed over a pervasive cluster, initially presented in (Steffenel et al., 2016). This application implements a surveillance and alert system for ultra-violet risks due to ozone-layer events all while relying on already existing computational resources, representing no additional cost to the institution.

Indeed, it is well known that the discovery of the Antarctic Ozone hole (Farman et al., 1985) raised the interest of the scientific community, with several studies monitoring the variation in the density of the ozone layer on polar regions (Salby et al., 2012). Inhabited zones can also be affected

due to both movements of the polar vortex borders over these regions (Marchand et al., 2002) or by the influx of air masses with reduced concentration of ozone detached from the polar vortex. The latter case, known as Ozone Secondary Effects (OSE), may cause a temporary reduction in the total column ozone (TCO) over populated areas (Manney et al., 1994; Pinheiro et al., 2011) and trigger important public health subjects as a reduction of 1% in the total ozone column may lead to an increase of 1.2% in the ultraviolet radiation (Guarnieri et al., 2004).

Today OSE can only be detected *a posteriori*, as satellite and ground instruments can at most inform the current status of the TCO over an area. To create an effective monitoring and public health alert system, we need to forecast OSE. Unfortunately, most of the climatic models are limited to the lower atmosphere layers (especially those associated to the weather forecast), and do not explore the interactions in higher layers, like those associated to the Ozone layer.

Instead of adapting a simulation model, our approach to forecast OSE relies in the analysis of historical data, looking for correlation patterns that can lead to a good forecast. Using CloudFIT, we created a pervasive HPC platform out of existing computational resources, minimizing the operational costs as there is no need to investment in the acquisition of dedicated machines or in the leasing of cloud resources.

Experiment Methodology

In the case of the Ozone Secondary Events detection, the problem was defined as a set of different tasks to be performed, each one exploring the CloudFIT distributed computing environment to its advantage. These tasks are described in Table 1.

Each step can be parallelized and the dependency between different jobs can be represented as a DAG, as illustrated in Figure 4. Furthermore, we can set different communities for different parts of the workflow. For instance, low-entry devices gathered at community C1 can be used to pre-process and store the data from ground-based Brewer or Dobson spectrophotometers as well as those readings from the OMI sensors from the satellites.

The newly entered data can trigger the OSE detection procedure, or in the case of a deeper analysis, launch an historical search for recurrent patterns that could be useful for event forecasting. This step, covering both filtering and time series analysis, requires more computing resources, so a community C2 composed by more powerful nodes can ensure this work.

Figure 4 also includes two other communities, C3 and C4, used to perform the last steps associated to the detection and forecasting of OSE.

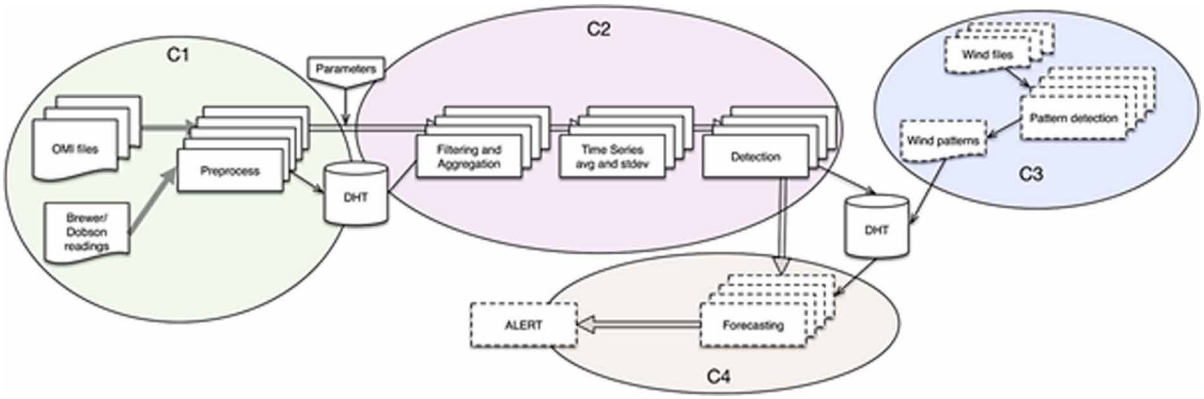
Input Preprocessing

The total ozone column can be measured by ground equipment but also counts with a worldwide satellite coverage, the OMI instrument, which produces a global measurement once a day. TOMS/OMI website offers different datasets, from raw data to final analyzed products. In our case, we use access raw data to extract all the necessary information to our calculations. The file format provided

Table 1. Detail of OSE analysis steps

Step	Actions
Input preprocessing	transform raw OMI data files for the analysis
Filtering and aggregation	select data corresponding to a given geographical zone and time window, performing aggregation if needed
Time series analysis	extract of correlation parameters for the target period and zone
Event detection	identification of abnormal ozone values and eventual alert generation
Event forecast	application of parameters to forecast OSE over inhabited areas

Figure 4. Workflow for the OSE detection framework



by OMI, presented in Figure 5 (at the left side) is not really adapted for parallel processing. Each file corresponds to the measures of a day and contains a header that specifies the basic information for that file (date, coordinates grid, step), followed by the measurements for each latitude (indicated at the end of the line) and for all covered longitudes. Each measure in Dobson Units (UD) is represented as a 3 digits integer that must be parsed from the other readings. For example, in Figure 5, the coordinates (-89.5, -179.5) has a value 280, (-89.5, -178.5) has a value 280, and so on.

As this format is difficult to manipulate, we preferred to preprocess it and store the data as JSON entries using the template presented in Figure 5 (at the right side). JSON is a well-known format that is easy to parse and also compatible with document-oriented NoSQL databases. The preprocessing is highly parallelizable as each input file containing the measures of a given day can be treated independently from each other and distributed over all the nodes using the DHT.

A similar procedure is implemented for other atmospheric parameters like wind currents. NetCDF files containing U and V components of wind currents at different altitudes are transformed in JSON entries for each coordinate pairs. As a result, the analysis of TCO concentration and dominant wind currents can be performed in the next task.

Time-Series Analysis, OSE Detection and Forecast

As presented above, OSE are defined by abnormal drops in the total ozone column that are not directly related to the expansion of the Antarctic Ozone Hole. The detection of the ozone concentration reduction is made by comparing the measured values for a given day with and the historical average for that zone. The current approach used by Vaz Peres (2013) relies on the historical average by month, i.e., measures are compared to the corresponding month average. For example, the October 22, 2013 values would be compared to the historical average for the October month. While this approach allows the detection of OSE, it lacks precision as the natural ozone concentration varies from year to year and also from the beginning to the end of the month, especially in transition months like July or

Figure 5. OMI Ozone raw data file for a given day and its JSON representation

```
Day:   1 Jan  1, 2013    OMI TO3    STD OZONE    GEN:13:003 Asc LECT: 01:44 pm
Longitudes:  360 bins centered on 179.5  W  to 179.5  E   (1.00 degree steps)
Latitudes :  180 bins centered on  89.5  S  to  89.5  N   (1.00 degree steps)
280280280280280280280280280280280280280280280280280279279279279279279279
279279279279279279279279279279279279279279279279279279279279279279279279
279279279279279279279279279279279279279279279279279279279279279279279279
279279279279279279279279279279279279279279279279279279279279279279279279
280280280280280280280280280280280280280280280280280280280280280280280280
280280280280280280280280280280280280280281281281281281281281281281281281
(...)
282282282282282282282282282282   lat =  -89.5
(...)   lat =  -88.5
(...)   lat =  -87.5
(...)
```

```
{
"date":"20130101",
"step":"1.0",
"latitudes":{
"-89.5":["-179.5":"280","-178.5":"280",(...)],
"-88.5":["-179.5":"272","-178.5":"272",(...)],
(...)
}
}
```

November. Using a standardized average for each month would result in false-positive alerts that we wish to minimize. Therefore, in order to make more precise estimations, our implementation uses a sliding window approach that computes averages and standard deviations for a time series of 15 days. This solution is more realistic as it allows natural variations in a given period to be taken into account.

Once the average and standard deviation for a given coordinate (in a given time period) is computed, we can proceed with the detection of OSE. One easy formula, applied on the experiment presented in the next section, considers the occurrence of an OSE if the measure for a given coordinate is less than the expected lower bound (1.5× the standard deviation) with respect to the last 15 days average.

While OSE detection is important, to implement an OSE forecast model is a major objective of this project. As developing a specific atmosphere modeling for the Ozone layer is beyond our scope, we decided to implement forecasting by detecting recurrent patterns leading to OSEs. For this, an extensive correlation analysis between OSE occurrences and atmospheric factors like wind currents must be performed. Hence, the use of different communities on CloudFIT allows the computation of an historical correlation database, all while online OSE detection/forecast is performed on the most recent Ozone readings. We believe that with such approach even a single low-end machine can ensure a daily forecast report, relying on patterns previously computed by a large community on the CloudFIT network.

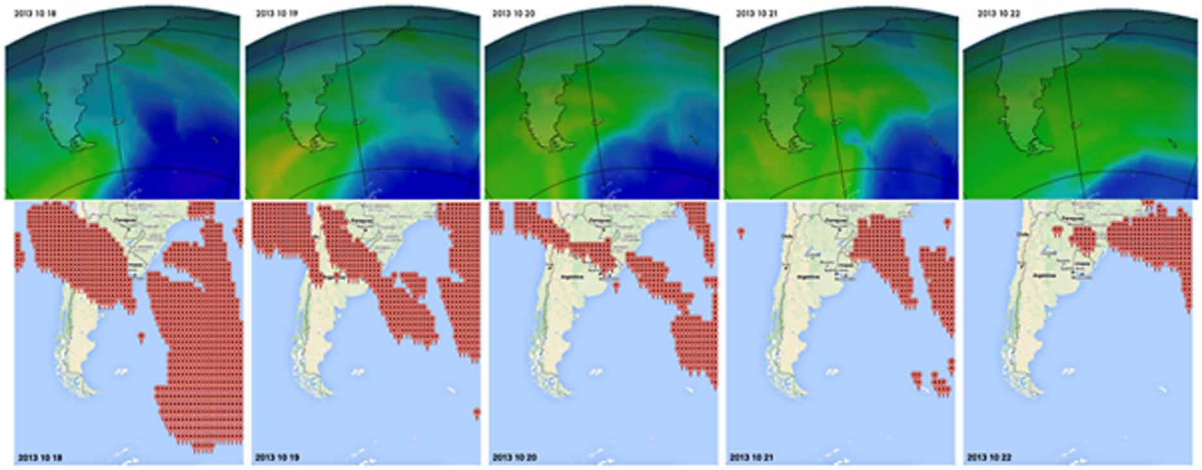
Experimental Results

In order to evaluate the effectiveness of deploying the OSE detection application on top of CloudFIT, we considered the first analysis steps (preprocessing to OSE detection) for the period between October 15th, 2013 and October 31st, 2013 over different devices, both individually and in a pervasive cluster mode. This scenario considers an area spanning 50°x55° (2750 coordinate points) on the south of South America that coincides with the area covered by the analysis from Vaz Peres (2013) and used to validate our results. For instance, in Figure 6, the upper part contains the Total Column Ozone (TCO) concentration from October 18th to October 22, 2013 over the South Pole according to the NASA *Ozone Hole Watch* Web site (http://ozonewatch.gsfc.nasa.gov/), while the lower part shows the progression of the OSE front as detected by our framework. The main advantage of our method is that instead of simply observing the Ozone concentration we are able to highlight the zones that experience a TCO drop, emphasizing the impact of OSE and allowing a more precise evaluation of the health risks.

Also, in order to evaluate the execution performance of our algorithms, we conducted the same experiment over three different platform configurations. In the first one, 3 Raspberry Pi 2 (900MHz 4-core ARM Cortex-A7, 1GB RAM) collaborate in a small network; in the second configuration, we created a heterogeneous network composed by one Raspberry Pi 2, one Macbook Air (Intel i7-4650U, 2 cores, 8GB RAM) and one Lenovo U110 (Intel Core2 Duo). Finally, the algorithm was run in a single Dell Precision T5610 server (Intel Xeon E5-2620, 12 cores, 32GB RAM). The first two configurations aim to represent edge/pervasive computing platforms that could be found in a real situation: the pure Raspberry Pi network may represent the low-end devices that serve as a first computing layer to the edge computing network, while the heterogeneous network represents a voluntary computing network composed by devices available at a given moment. The Xeon server, on the other hand, is used as a control reference to infer the performance of the algorithm in a more classical infrastructure.

Hence, the analysis of the OMI data from satellites in the Raspberry Pi 2 network took around 30 minutes, while the pervasive cluster required about 10 minutes. By comparison, the Xeon server required less than 7 minutes. Furthermore, in the case the application shall perform a global coverage (65000+ coordinates) or requires more data intensive operations (historical analysis, forecast), new nodes can join CloudFIT at any moment and thus provide the elasticity required to perform the tasks under different loads.

Figure 6. Absolute Ozone concentration and the OSE progression between October 18 to October 22, 2013



CONCLUSION

The digital economy that marked the turn of this century changed in deep the way companies plan and manage their business models, and big data analytics is one of the major tools supporting this revolution. It is an error, however, to think that big data requires massive investments and the usage of large-scale resources like those found in a dedicated datacenter or in a cloud computing facility. Some organizations are more prone to lightweight and less expensive platforms, we strongly believe that these organizations can boost their efficiency with minimal costs by relying on edge computing over pervasive grids. For this reason, this paper shows how edge computing and pervasive grids can be used to develop efficient and flexible big data analytic tools and applications.

To illustrate this approach, we present a distributed computing platform that implements the concepts of both edge computing and pervasive grids. We demonstrate its utilization to deploy an atmospheric surveillance system developed for a governmental institution.

The works presented in this paper represent only the first steps towards the generalization of big data on edge and pervasive environments. Indeed, additional research concerning context awareness is essential, as several advances can be obtained through it: better usage of resources, energy awareness (green computing), proximity services, etc.

It is worth noting that the increasing number and nature of proximity devices (IoT, smartphones, etc.) represents an unprecedented computing power at the reach of our hands, and that edge and pervasive computing are tools that can help unleash that computing power. Indeed, efficiently exploring these new resources is a valuable asset for any organization, both in cost reductions and in the sustainable usage of the resources.

REFERENCES

Anderson, D. (2004) BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. doi:10.1109/GRID.2004.14

Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., & Werthimer, D. (2002). SETI@home: An experiment in public-resource computing. *Communications of the ACM*, *45*(11), 56–61. doi:10.1145/581571.581573 PMID:12238525

Apache Hadoop. (2016). Retrieved from http://hadoop.apache.org/

Babiceanu, R. F., & Seker, R. (2016). Big Data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook. *Computers in Industry*, *81*, 128–137. doi:10.1016/j.compind.2016.02.004

Bocek, T. (2015). TomP2P, a P2P-based high performance key–value pair storage library. Retrieved from https://tomp2p.net/

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the 1st MCC Workshop on Mobile Cloud Computing, MCC '12*, New York, NY. doi:10.1145/2342509.2342513

Brasileiro, F., Araujo, E., & Voorsluys, W., Oliveira & M. Figueiredo, F. (2007) Bridging the High Performance Computing Gap: the OurGrid Experience. *7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)* (pp. 817-822). doi:10.1109/CCGRID.2007.28

Cassales, G. W., Charao, A., Kirsch-Pinheiro, M., Souveyet, C., & Steffenel, L. A. (2016). Improving the performance of Apache Hadoop on pervasive environments through context-aware scheduling. *Journal of Ambient Intelligence and Humanized Computing*, *7*(3), 333–345. doi:10.1007/s12652-016-0361-8

Celaya, J., & Arronategui, U. (2011) A Highly Scalable Decentralized Scheduler of Tasks with Deadlines. In *Proceedings of the IEEE/ACM 12th International Conference on Grid Computing* (pp. 58-65).

Cisco. (2013). *Cisco Research Center Requests for Proposals (RFPs)*. Retrieved from http://research.cisco.com/research#rfp-2013078

Dey, S., Mukherjee, A., Paul, H. S., & Pal, A. (2013). Challenges of using edge devices in IoT computation grids. In *Proceedings of the Int. Conf. on Parallel and Distributed Systems* (pp. 564–569). doi:10.1109/ICPADS.2013.101

ETSI. (2014). *Mobile-edge computing - introductory technical white paper*. Retrieved from http://bit.ly/2bzLQ8m

Farman, J., Gardiner, G., & Shanklin, J. (1985). Large losses of total ozone in Antarctica reveal seasonal clox/nox interaction. *Nature*, *315*(6016), 207–210. doi:10.1038/315207a0

Garcia Lopez, G., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., & Riviere, E. et al. (2015). Edge-centric computing: Vision and challenges. *Computer Communication Review*, *45*(5), 37–42. doi:10.1145/2831347.2831354

Gartner. (2011, June 27). *Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data*. Newsroom. Retrieved from http://www.gartner.com/newsroom/id/1731916

Guarnieri, R., Padilha, L., Guarnieri, F., Echer, E., Makita, K., Pinheiro, D., & Schuch, N. et al. (2004). A study of the anticorrelations between ozone and UV-B radiation using linear and exponential fits in southern Brazil. *Advances in Space Research*, *34*(4), 764–776. doi:10.1016/j.asr.2003.06.040

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*(January), 98–115. doi:10.1016/j.is.2014.07.006

Jagadish, H., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big Data and Its Technical Challenges. *Communications of the ACM*, *57*(7), 86–94. doi:10.1145/2611567

Johnen, C., & Mekhaldi, F. (2011). Self-stabilization versus robust self-stabilization for clustering in ad-hoc network. In *Proceedings of the 17th International Conference on Parallel Processing* (pp. 117-129). Springer. doi:10.1007/978-3-642-23400-2_12

Krajecki, M., & Jaillet, C. (2004). Solving the Langford problem in parallel. In *Proceedings of the 3rd International Workshop on Parallel and Distributed Computing* (pp. 83-90). IEEE.

Lim, L., & Conan, D. (2014). Distributed event-based system with multiscoping for multiscalability. In *Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing*. ACM. doi:10.1145/2676733.2676736

Manney, G. L., Zurek, R. W., O'Neill, A., & Swinbank, R. (1994). On the motion of air through the stratospheric polar vortex. *Journal of the Atmospheric Sciences*, *51*(20), 2973–2994. doi:10.1175/1520-0469(1994)051<2973:OTMOAT>2.0.CO;2

Marchand, M., Bekki, S., Pazmino, A., Lefèvre, F., Godin-Beekmann, S., & Hauchecorne, A. (2005). Model simulations of the impact of the 2002 Antarctic ozone hole on the midlatitudes. *Journal of the Atmospheric Sciences*, *62*(3), 871–884. doi:10.1175/JAS-3326.1

Mora, M., Gelman, O., Paradice, D., & Cervantes, F. (2008). The case for conceptual research in information systems. In Proceedings of the CONF-IRM 2008 (p. 52).

Pahl, C., & Lee, B. (2015) Containers and Clusters for Edge Cloud Architectures – a Technology Review. In *Proceedings of the 3rd IEEE International Conference on Future Internet of Things and Cloud (FICloud)* (pp. 379-386). doi:10.1109/FiCloud.2015.35

Parashar, M., & Pierson, J. M. (2010). Pervasive grids: Challenges and opportunities. In K. Li, C. Hsu, L. Yang et al. (Ed.). Handbook of Research on Scalable Computing Technologies (pp. 14–30). Hershey, PA: IGI Global.

Peres, L. V., Bencherif, H., Mbatha, N., Schuch, A. P., Toihir, A. M., Bègue, N., & Schuch, N. J. et al. (2017). Measurements of the total ozone column using a Brewer spectrophotometer and TOMS and OMI satellite instruments over the Southern Space Observatory in Brazil. *Ann. Geophys.*, *35*, 25–37. doi:10.5194/angeo-35-25-2017

Pinheiro, D., Leme, N., Peres, L., & Kall, E. (2011). *Influence of the Antarctic ozone hole over South of Brazil in 2008 and 2009*. National Institute of Science and Technology.

Ramakrishnan, A., Preuveneers, D., & Berbers, Y. (2014). Enabling self-learning in dynamic and open IoT environments. In *Proceedings of the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)* (pp. 207–214).

Salby, M. L., Titova, E. A., & Deschamps, L. (2012). Changes of the Antarctic ozone hole: Controlling mechanisms, seasonal predictability, and evolution. *Journal of Geophysical Research, D, Atmospheres*, *117*(D10). doi:10.1029/2011JD016285

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14–23. doi:10.1109/MPRV.2009.82

Shekhar, S., & Gokhale, A. (2017) Dynamic resource management across cloud-edge resources for performance-sensitive applications. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. doi:10.1109/CCGRID.2017.120

Steffenel, L. A., & Kirsch-Pinheiro, M. (2015). When the cloud goes pervasive: approaches for IoT PaaS on a mobiquitous world. In *Proceedings of the EAI International Conference on Cloud, Networking for IoT systems (CN4IoT 2015)*.

Steffenel, L. A., Kirsch-Pinheiro, M., Kirsch-Pinheiro, D., & Vaz Peres, L. (2016). Using a Pervasive Computing Environment to Identify Secondary Effects of the Antarctic Ozone Hole. In *Proceedings of the 2nd Workshop on Big Data and Data Mining Challenges on IoT and Pervasive (Big2DM)*. doi:10.1016/j.procs.2016.04.215

Vaz Peres, L. (2013). *Efeito Secundário do Buraco do Ozônio Antártico Sobre o Sul do Brasil* [Msc in Meteorology dissertation]. Universidade Federal de Santa Maria, Brazil.

Vermesan, O., Friess, P., Guillemin, P., Giaffreda, R., Grindvoll, H., Eisenhauer, M., & Tragos, E. Z. et al. (2014). Internet of Things beyond the hype: Research, innovation and deployment. In *Internet of Things - From Research and Innovation to Market Deployment*. River Publishers.

Wang, W., Barnard, M., & Ying, L. (2015). Decentralized scheduling with data locality for data-parallel computation on peer-to-peer networks. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL. doi:10.1109/ALLERTON.2015.7447024

Wright, A. (2014). Big Data Meets Big Science. *Communications of the ACM*, *57*(7), 13–15. doi:10.1145/2617660

Wu, D., Tian, Y., & Ng, K.-W. (2005) Aurelia: Building locality-preserving overlay network over heterogeneous P2P environments. In *Proceedings of the International Conference on Parallel and Distributed Processing and Applications*. Springer.

Luiz Angelo Steffenel is Associate Professor at the University of Reims Champagne-Ardenne, France. He obtained a Ph.D. in Computer Science in 2005 at Institut National Polytechnique de Grenoble, France. Dr Steffenel is a board member of the French Grid'5000 project. He is also its scientific coordinator for the University Reims Champagne-Ardenne. His research interests include parallel and distributed systems, grid computing, fault tolerance and pervasive computing. Also, he works on parallel approaches for molecular docking, in collaboration with researchers from Institut de Chimie Moléculaire (ICMR/CNRS) and Matrice Extracellulaire et Dynamique Cellulaire (MEDyC/CNRS).

Manuele Kirsch Pinheiro is Associate Professor in the Computer Science Research Center (Centre de Recherche en Informatique) of the University of Paris 1 Panthéon-Sorbonne. Previously, she occupied a post-doctoral position on the Computer Science of the Katholieke Universiteit Leuven. She received her PhD in computer science from the University Joseph Fourier – Grenoble I in 2006, Grenoble, France. Her research interests include ubiquitous computing, context-aware computing, pervasive grids, pervasive information systems, cooperative work (CSCW) and group awareness.

Lucas Vaz Peres has a PhD in Meteorology from Universidade Federal de Santa Maria (2016). Works in the field of Geosciences, focusing on Meteorology, with a special interest for the following subjects: synoptic analysis, brewer metering, the Antarctic ozone hole and its secondary effects, and the optical thickness of aerosols.

Damaris Kirsch Pinheiro is PhD in Space Geophysics (2003). She was Director of the Space Science Laboratory of Santa Maria- LACESM/UFSM and she is the coordinator of the Chemical Engineering undergraduate program at Universidade Federal de Santa Maria. Dr Kirsch Pinheiro represented the Brazilian government at UNEP in the 9 and 10 Ozone Research Meetings.