Data Movement in the Internet of Things Domain

Francesco D'Andria[1], Daniel Field[1], Aliki Kopaneli[2], George Kousiouris[2], David Garcia-Perez[1], Barbara Pernici[3], and Pierluigi Plebani[3(✉)]

[1] Atos Spain SA, Avenida Diagonal 200, 08018 Barcelona, Spain
{francesco.dandria,daniel.field,david.garcia-perez}@atos.net
[2] Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str., 15773 Athens, Greece
{alikikop,gkousiou}@mail.ntua.gr
[3] Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy
{barbara.pernici,pierluigi.plebani}@polimi.it

Abstract. Managing data produced in the Internet of Things according to the traditional data-center based approach is becoming no longer appropriate. Devices are improving their computational power as the processors installed on them are more and more powerful and diverse. Moreover, devices cannot guarantee a continuous connection due their mobility and limitation of battery life.

Goal of this paper is to tackle this issue focusing on data movement to eliminate the unnecessary storage, transfer and processing of datasets by concentrating only the data subsets that are relevant. A cross-layered framework is proposed to give to both applications and developers the abstracted ability to choose which aspect to optimize, based on their goals and requirements and to data providers an environment that facilitates data provisioning according to users' needs.

Keywords: Data movement optimization · Internet of Things · Information and data quality

1 Introduction

Customised and Low Power Computing is a disruptive innovation which, while immediately of benefit to the established datacentre-centric model of the current IT world, opens the gate to many fields where datacentre-based computing is simply not appropriate. Predictions abound for the IoT, the Smart Everything Anywhere initiative and cyber-physical systems. In this scenario, applications must deal with the data deluge produced by this increasing amount of devices. To properly manage these big amounts of data, their quality needs to be certified, so the performance of mechanisms to access them must satisfy the developer requirements.

The goal of this paper is to propose a conceptual framework that tackles these issues as a basis to support the data movement optimization: how to select

S. Dustdar et al. (Eds.): ESOCC 2015, LNCS 9306, pp. 243–252, 2015.
DOI: 10.1007/978-3-319-24072-5_17

what data to transfer, when and to where, in order to reduce the requirement for storage capacity and data processing capacity (reducing hardware requirements) and increasing the speed of producing meaningful output (performance), while guaranteeing data preservation where required. The approach is based on a cross-layer optimization framework enabling the data movement by introducing *data curators* as responsible for storing and managing data coming from the devices according to models able to match the data consumer requirements and the data providers capabilities. A programming abstraction is also proposed to ease the customization of devices and to hide the underlying mechanisms needed to access and to move the data in order to satisfy requirements on performances, data quality, energy, and security.

The rest of the paper is structured as follows. Section 2 motivates the need for data movement in IoT scenario. In Section 3 the envisioned cross-layer optimization framework is presented. Section 4 discusses a possible solution for managing the data movement optimization, while Section 5 introduces the metamodel supporting the programming abstraction that is at the basis of the framework. Finally, after a discussion on related work in Section 6, Section 7 concludes the paper outlining future extensions.

2 Motivation

IDC estimates that the digital universe will reach the size of 40ZettaBytes in 2020 [4] and IoT holds a significant role in this information deluge. It is now crucial to find methods and tools for making accessible these big data to the right users, at the right time, with the right quality, and easily. At the same time, customized and low-power computing devices will constitute the majority of these things, and their ever-increasing computational power opens the possibility to access to a potentially unlimited set of processors. Data movement optimization can take advantage of this situation by pushing part of the computation for optimization directly to the device in order to reduce the unnecessary storage, transfer and processing of datasets.

In a typical situation, data producers are responsible of managing and configuring devices, e.g., wearables, sensors, smartphones, single board computers, that produce data and that can be seen as data sources. Each of them has its own characteristics in terms of energy autonomy, storage size, and communication interface. On the other side, data consumers are responsible for developing and maintaining applications that need the data produced by devices. As the computational power of devices is continuously increasing, this approach does not really exploit all the potential at data sourcing layer, as the processors installed on the devices are not fully used. At the same time, due to the limitation in terms of storage and autonomy of energy, computation to be executed on the device needs to be properly balanced with respect to such limitations. Moreover, device storage limitation requires that data should move from more volatile storage systems (at device level) to more persistent ones, if data must be available for a longer period, for instance, for analysis purposes or for satisfying preservation requirements.

Data movement optimization is thus required to move data from devices that own fresh data - but limited in size - to more capable storage systems affordable only at data consumption layer, leaving the possibility for the devices to process data exploiting their computational power in order to reduce the amount of data transmitted to what is really useful for the user.

Deciding when, where, and how to move data between data sourcing layer and data consumption layer is not an easy task as it depends on several aspects. First of all, latency strictly depends on the usage: for some users, (e.g., real time applications) having timely data is crucial, for some other users (e.g., data analytics) even not so up-to-date data are considered sufficient. Data granularity depends on the user needs and the device should support the possibility of being configured to sample with different frequencies. Devices could not natively support data encryption, but users could request for an encrypted transmission, so it should be possible to install and run encryption modules on the device exploiting the processors on board. Last but not least, battery autonomy may influence the duration of the transmission and the proper amount of data to be transmitted should be calculated.

3 The Cross-Layer Optimization Framework

To be able to support data movement, taking into account the open issues discussed above, the presented approach enriches the data movement eco-system (see Figure 1) introducing data curators as key-players that mediate between the capabilities offered by the devices and the requirements posed by the consumer. Moreover, thanks to the metamodel supporting programming abstraction, the proposed approach eases the distribution of data processing among the layers to exploit the computational power now available at device layer also considering devices' limited resources.

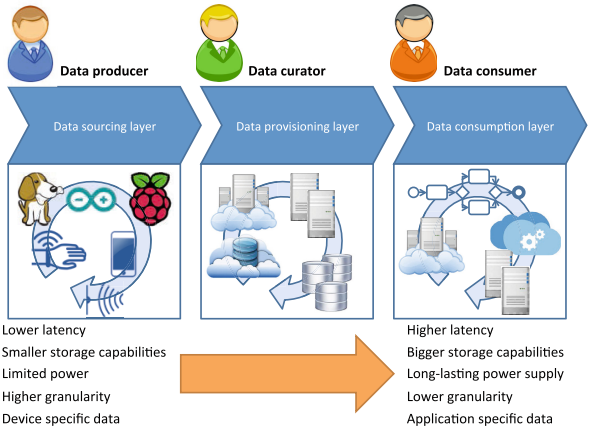

Fig. 1. Enriched data movement eco-system.

The data provisioning layer improves what the data sourcing layer offers with the possibility of additional computing power and storage, improving the quality of data provided in terms of accuracy, precision, and timeliness [11]. The higher the quality, the higher the value of the data provided and, at the same time, the higher the quality, the higher the amount of resources required, as well as costs. It is worth noting that although the role of the data curator is fundamental for having ready-to-use data with certified level of quality, with the proposed eco-system data consumers can keep accessing directly the data producers to obtain data, but without the possibility to have access to the services provided by the data curator. Moreover, the data movement can occur not only between layers, but also among peers referring to the same layer. For instance, data movement between embedded systems can be considered as a possible scenario, as well the migration of data between storage nodes living on two different platforms.

Figure 2 shows the envisioned cross-layer optimization framework supporting the enriched data movement eco-system. Each layer offers the possibility to access to the data specifying the data needed and the *goals* corresponding to the non-functional requirements (e.g., high accuracy, low energy consumption). Goals can be hard (must be satisfied) and soft (should be satisfied). To satisfy the request, each layer enables the data access according to one or more *modes*, each of them having different impacts in terms of performance, data quality, energy, and security [8]. For instance, compressed data involve more computation than uncompressed one, but it reduces the time for transmission. Secured data transmission ensures the integrity and confidentiality, but introduces an overhead before transmitting and after receiving data.

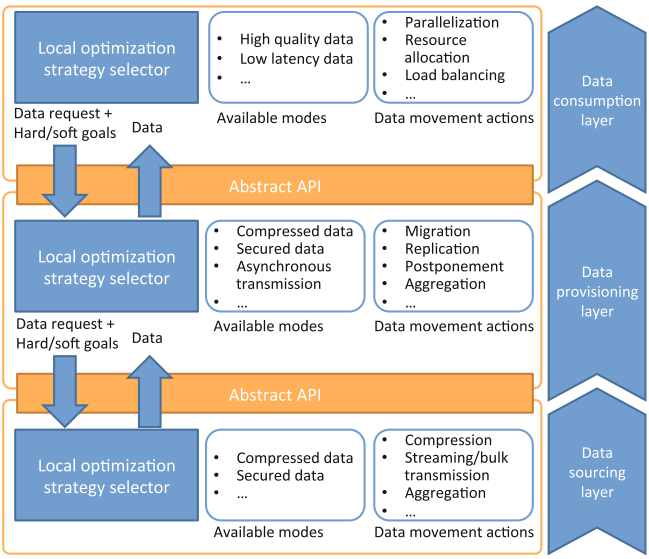

Fig. 2. Overall architecture

Data movement optimization is achieved at each layer by finding the best match between goals and modes considering that different users access the same data with different goals and according to different modes. Finding this match is the main task of the *local optimization strategy selector*, that identifies the best mode according to the user request and the current resource usage and exploiting the available *data movement actions*: layer-specific actions whose objective is to move data inside a layer or across layers to better support the data offering. For instance, at data sourcing layer, data movement actions concern the device configuration to support the transmission of compressed data or the aggregation of samples to reduce the amount of data to be transmitted.

As the data movement optimization may require some computation inside a layer to enable the execution of a mode, the proposed approach aims to exploit the full computing potential that devices installed can provide. At data sourcing layer, computing power of devices is continuously increasing and, usually, all the potential that the on-board processors can provide is not fully exploited. At data provisioning layer, servers install Advanced Processing Units (APU), such as GPUs, that can be used to execute the processing required by the data movement actions. For instance, if a device is not capable to compress data but compression is required, capabilities of the device can be extended by deploying code compliant to frameworks such as OpenACC or OpenCL, to ensure code portability. Such an extension can be directly done by the device owner when it has the skill to develop it, or even by the data curator that deploys the code to the device using the programming abstraction.

Data movement actions can also occur inside the same layer involving different peers exchanging data to better support the user request. Parameters as data proximity and data locality can be defined to move the relevant data set closer to the place in which the data will be used. For instance, at data provisioning layer, a data curator can decide to replicate a data set on different geographically distributed storage nodes to better support worldwide access. In this case, the data movement action does not have a correspondent mode, but its enactment will have an impact on the data access performance.

4 Optimizing Data Movement

To make explicit the relationship among the data movement actions, the modes and the goals are correlated through a data-movement/effects relationship map. This map defines the dependencies among the quality associated to data that a source can offer with respect to the effort required to provide those data. This map will include different types of indicators such as: (i) performance indicators that measure the efficiency of data manipulation (e.g., response time); (ii) data quality indicators that measure the effectiveness of applications (e.g., data accuracy); (iii) energy indicators that measure the resource impact of data accessing and storing (e.g., energy consumption per unit of work or per Kbyte); (iv) security indicators that define the level of trust, confidentiality, and integrity of the data. Dependencies between the goals and modes with the indicators will be

defined to identify the effects on enacting a mode or requesting a goal in terms of data quality, energy, efficiency and security.

The indicators map constitutes the basis for this goal-based model, as it provides a way to specify the as-is and to-be scenarios of the system before and after the enactment of the data movement actions in terms of performance, data quality, energy, and security aspects. To pursue a data movement optimization all these variables need to be managed in a coherent way and their mutual dependencies need to be properly mapped. To this aim, goal-based models can be adopted to evaluate the impact of the data movement actions enacted [1].

Since it is difficult in the general case to design at run time all the possible relationships between indicators, learning tools, such as Bayesian Networks, can be used to derive the relations among the indicators and the goals, and to evaluate the effect of selecting a mode to provide data on these indicators [14].

A possible approach for performing dynamic correlations in service-based environments is discussed in [7]. In our case the models would be adapted as it appears in Figure 3. Available modes can be numerically encoded and introduced as inputs, along with device IDs, to indicate the used hardware. Inputs can also be considered any operation specific aspects that may affect e.g. the workload of the operation, like data size to be moved etc. The outputs should be the observed (in the case of training) or predicted effects on the defined indicators for the various goals (e.g., energy levels, performance timings etc.). Such observations may also be acquired at run-time through relevant annotations in the implemented modes, that may inject code to monitor the specific metrics. Through training with historical data the effect analysis models may be created and used a priori for optimized management and mode selection.

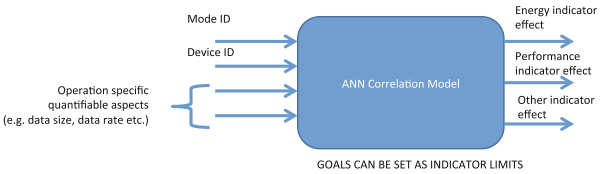

Fig. 3. Annotation correlation model

The role of the data-movement/effect relationship map is crucial in the data movement optimization at both local and global level. At local level, the map supports the data provider to identify the best mode to be proposed to satisfy the user goals and the possible effects of a data movement action on the goal satisfaction. At global level, the map provides a shared view for all the local decision-makers to understand the cross-layer effects of their local data movement actions. Furthermore, the effect of a specific mode (e.g., speed up factor following a parallelization pattern) may be quantified.

5 Metamodel Supporting the Programming Abstraction

To reduce the complexity for data curators and data consumers when accessing and manipulating data, a programming abstraction is offered. Data access provided by each layer will be offered through a single abstracted interface based on modes and goals: modes correspond to how data may be offered (e.g., compressed, aggregated), whereas goals correspond to the non-functional requirements (e.g., high accuracy, low energy consumption).

The programming abstraction is based on a metamodel to represent data sources characteristics, data consumer's requests, and related requirements. The metamodel is composed of the following elements, illustrated with some exemplifications:

- *data sources* : capabilities (for security, compression); resources (power usage, available storage, available main memory); layer (sourcing/provisioning)
- *data elements* description; characteristics:
  - data modes: compressed, aggregated (with parameters);
  - data properties: freshness/time span, update rate, locality, quality (accuracy)
- *data requests*: data elements with requested data modes and data properties; transmission mode (compressed/not compressed; secured/not secured; streamed/bulk); goals (energy efficiency, time constraints, ...)

Developers, when coding applications consuming the data, decorate with annotations the methods that access to the data with the goals (e.g., energy efficiency) to be satisfied. The local optimization strategy selector implements the logic that is able to identify the best source and the best mode to be invoked to satisfy the goal. In the example shown in Figure 4 , two possible sources could satisfy the request: one directly connected to the devices (for more recent data), another managed by a data curator that periodically collects the data (e.g., once a day, not shown in the figure) from the device and stores them into a permanent storage system to guarantee a longer persistence, but not providing up-to-date information. When the application requires for today's data, the abstract request is automatically instantiated to invoke the *getUncompressedData* offered by the data producer abstract API. If requested data are less recent, then two possibilities are open, i.e., with or without compression and the selection will be performed to satisfy the consumers goal.

To cope with additional requirements, the programming abstraction also supports the extension of capabilities provided by the devices. For instance, the developer explicitly asks for today's positions in a compressed format and with specific performance requirements in terms of data acquisition delay. As the device is the only source for this data, the possibilities for the selection are three: reading all the data uncompressed and compress them locally, deploy a standard compression module on the device to augment the available modes so that data can be sent already compressed or deploy an OpenCL implementation exploiting potential multicores (e.g. in a mobile device). Also in this case, the decision will depend on the trade-off analysis of the impact in choosing the best

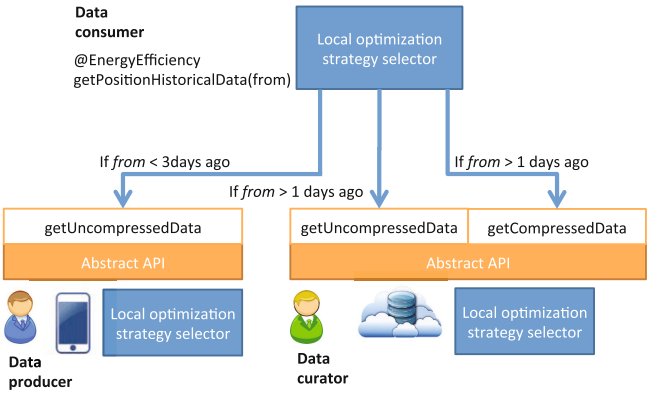

Fig. 4. Data movement strategy selection.

option considering that: initially, the more computation done at device layer, the more energy will be consumed with the risk of running out of power, secondly the use of OpenCL may guarantee performance requirements in terms of task completion or finally that the less data transmitted, the less energy consumed.

6 Related Work

The interest around the data movement topic is gaining more and more attention in the research community especially because of the diffusion of sensors and smart objects and the potentially unlimited computational power provided by Cloud solutions increased the amount of available data. As a consequence, data consumers have the possibility to access an enormous data set and proper mechanisms to optimize this access are required. Data movement techniques aim to achieve this goal especially focusing on moving data at the right time to the right place, where time and place refers to the user that needs the data. Bulk data transfer protocols [12] are usually adopted to quickly transfer massive amounts of data between storage systems geographically distributed but they do not solve the problem, as the applications that need these data are highly dynamic and data movement could be required frequently and with different properties. Live data migration aims to cover this space, and the work done in [13] goes in this direction although it focuses on an environment managed by a single entity. Data replication is another possible technique involved in the data management. Usually adopted in cloud computing domain to bring data closer to the user, data replication may have impact on the energy consumption: the more the replicas the higher the energy consumed. In this direction,[2] proposes an approach able to balance these two needs, i.e., replicas and low energy, relying on a specific power consumption model. At the same time, the gap between the I/O capacity and the computational power of the systems requires solutions for deciding which

data needs to be moved and if pre-processing is required before the transmission. Data compression and data reduction mechanisms are adopted to reduce the amount of data to be transmitted without compromising the data quality, but requiring a computational effort before and after the transmission occurs. Another solution that is more focused on the device level is proposed in [3] where data collected by a sensor network are pre-processed before sending them to the storage node to reduce the energy consumption of the sensor networks without affecting the data quality requirements.

The cross-layered framework proposed in this work is based on a programming abstraction approach, to offer tools that will enable developers to limit their knowledge of the underlying layers, while achieving increased functionality levels. Especially using annotations, several work papers demonstrated the possibility to require and enforce optimization at code-level. In [5] annotations are used in software libraries as an effort for enforcing domain specific optimizations applied on them. The whole task is enabled via an annotation engine implemented as a compiler for translating the annotation semantics into code optimization. Following the same path, in [10] annotations are used for passing high-level semantic information to the compiler, in order to overcome performance issues resulting from the use of abstractions. From a slightly different point of view, [9] uses annotations via a source-to-source translation system aiming at optimizing MPI applications while [6] utilizes annotations (and an implemented annotation engine) in order to increase the speed of collaborative web applications development.

7 Concluding Remarks

The amount of smart devices is significantly increasing and, although they have an important role in causing the so called data deluge, they are also an opportunity to better manage the produced data. In this paper, we have focused on this perspective on Big Data, proposing a cross-layer framework whose main goal is to optimize the data movement.

The proposed framework enriches the usual approach by introducing the data curator as the actor in charge of mediating between the capabilities offered by the devices and the requirements posed by data consumers. Moreover, the framework supports the data movement optimization providing a programming abstraction that eases the interaction with the devices enabling the data access and their configuration to better exploit the devices' computational power.

As the proposed framework constitutes a first attempt to exploit the devices' capabilities inside the Big Data domain, future work will better investigate the impact on data movement optimization at local (inside a layer) and global (among layers) level. Validation on a real scenario is also planned to quantify the efficiency and the effectiveness of the proposed approach.

References

1. Asnar, Y., Giorgini, P., Mylopoulos, J.: Goal-driven risk assessment in requirements engineering. Requir. Eng. **16**(2), 101–116 (2011)
2. Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A.Y.: Energy-efficient data replication in cloud computing datacenters. Cluster Computing **18**(1), 385–402 (2015). http://dx.doi.org/10.1007/s10586-014-0404-x
3. Cappiello, C., Schreiber, F.A.: Quality- and energy-aware data compression by aggregation in wsn data streams. In: Proc. of the 2009 IEEE Int'l Conf. on Pervasive Computing and Communications, PERCOM 2009, pp. 1–6. IEEE Computer Society, Washington, DC (2009)
4. Gantz, J., Reinsel, D.: Extracting values from chaos. IDC, June 2011. http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf
5. Guyer, S.Z., Lin, C.: An annotation language for optimizing software libraries. In: Proc. of the 2nd Conf, on Domain-specific Languages, DSL 1999, pp. 39–52. ACM, New York (1999). http://doi.acm.org/10.1145/331960.331970
6. Heinrich, M., Grüneberger, F.J., Springer, T., Gaedke, M.: Exploiting annotations for the rapid development of collaborative web applications. In: Proc. of the 22nd Int'l Conf. on World Wide Web, WWW 2013, Rio de Janeiro, Brazil, pp. 551–560 (2013)
7. Kousiouris, G., Kyriazis, D., Gogouvitis, S., Menychtas, A., Konstanteli, K., Varvarigou, T.: Translation of application-level terms to resource-level attributes across the cloud stack layers. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 153–160, June 2011
8. Kousiouris, G., Menychtas, A., Kyriazis, D., Konstanteli, K., Gogouvitis, S., Katsaros, G., Varvarigou, T.: Parametric design and performance analysis of a decoupled service-oriented prediction framework based on embedded numerical software. IEEE Transactions on Services Computing **6**(4), 511–524 (2013)
9. Nguyen, T., Cicotti, P., Bylaska, E., Quinlan, D., Baden, S.B.: Bamboo: translating mpi applications to a latency-tolerant, data-driven form. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC 2012, pp. 39:1–39:11. IEEE Computer Society Press, Los Alamitos (2012). http://dl.acm.org/citation.cfm?id=2388996.2389050
10. Quinlan, D., Schordan, M., Vuduc, R., Yi, Q.: Annotating user-defined abstractions for optimization. In: 20th International on Parallel and Distributed Processing Symposium, IPDPS 2006, p. 8, April 2006
11. Wang, R., Strong, D.: Beyond accuracy: What data quality means to data consumers. Journal of Management Information Systems **12**(4), 5–33 (1996)
12. Ren, Y., Li, T., Yu, D., Jin, S., Robertazzi, T., Tierney, B.L., Pouyoul, E.: Protocols for wide-area data-intensive applications: design and performance issues. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC 2012, pp. 34:1–34:11. IEEE Computer Society Press, Los Alamitos (2012)
13. Tai, J., Sheng, B., Yao, Y., Mi, N.: Live data migration for reducing sla violations in multi-tiered storage systems. In: Proceedings of the 2014 IEEE International Conference on Cloud Engineering, IC2E 2014, pp. 361–366. IEEE Computer Society, Washington, DC (2014). http://dx.doi.org/10.1109/IC2E.2014.8
14. Vitali, M., Pernici, B., OReilly, U.M.: Learning a goal-oriented model for energy efficient adaptive applications in data centers. Information Sciences (2015)