Back to the Future: Resource Management in Post-Cloud Solutions

Michele Zanella*, Giuseppe Massari*, Andrea Galimberti*, William Fornaciari*

ABSTRACT

The increasing pervasiveness of mobile and embedded devices (IoT/Edge), combined with the access to Cloud infrastructures, makes it possible to build scalable distributed systems, characterized by a multi-dimensional architecture. The overall picture is a massive collection of computing devices, characterized by very heterogeneous levels of performance and power consumption, which we could exploit to reduce the need to access Cloud computing resources. This would play a key role, especially for emerging use cases, where huge amount of data are generated. However, the deployment of distributed applications on such an heterogeneous infrastructures requires suitable management layers. This position paper aims at: (a) proposing a fully-distributed, cooperative, dynamic and multi-layered architecture, capable of integrating different computing paradigms; (b) identifying a possible solution to manage workloads at run-time in a resources continuity perspective, through an analysis of the open research challenges.

CCS CONCEPTS

1 INTRODUCTION

Both academia and industry have addressed the problem of the organization of computing systems and the increased demand of computational power, since a very long time. In this regard, looking back at the last decades, we can observe an interesting cyclic relationship, as shown in Figure 1.

Until the 1980s, centralized architectures dominated the enterprise scene, with mainframes and "dummy" terminals connected together, in a client-server fashion (Table 1). During the 1990s, the increase of performance of desktops and personal computers made it possible to decentralize the workload, arranging distributed systems, also in peer-to-peer (P2P) configuration, rather than client-server. In 2000s, *Cloud computing* decreed another turning point towards a return to centralized approaches, in which remote data and computing centers provide access to services and platforms in a global way. This regardless of users location or device's capabilities. Today, users and developers can in fact leverage on Cloud resources to deploy computing-intensive tasks, coming from Machine Learning and Deep Neural Networks or Big Data analysis applications. Moreover, users and companies can take advantage of large scale commercial services to decrease costs, by relying on ready-to-use Cloud services. Unfortunately, emerging use-cases are showing the limits of this approach [16, 24].

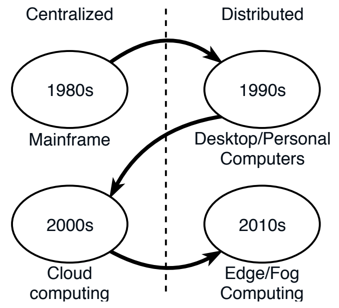


Figure 1: A cyclic view of computing approaches across ages.

According to Gartner, in 2016 there were 6,4 billions of edge devices around the world, whereas the estimation for year 2020 is of 20,8 billions. This exponential growth has pushed towards the *Internet of Things (IoT)* concept – the idea of having a plethora of physical devices (from sensors-based embedded systems to house appliances) interconnected via Internet. This paradigm allows the exploration of the pervasiveness of mobile and smart devices in users' daily life, collecting and exchanging a lot of data, for which different degrees of processing can be required. In this regard, novel application domains are arising in health care, smart cities, automotive and industry contexts.

However, the huge amount of raw data, combined with the high transfer rate,increases the network bandwidth requirements, with a consequent loss in terms of response times and network latency. This introduces the need of going beyond centralized Cloud computing based approaches. In this sense, it is worth noticing that not all the data are relevant for the final processing. The risk is to afford costs and spend Cloud resources to perform a not negligible amount of pre-processing operations. Furthermore, accessing Cloud infrastructures requires a stable low-latency Internet connection, which makes the exploitation of Cloud computing a challenging task for real-time applications.

*Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria <name.surname>@polimi.it.

Age	Approach	Implementations	Accessibility	Pervasiveness	Model
1980s	Centralized	Mainframe	Enterprise	Local enterprise context	Client-Server
1990s	Decentralized	Desktop/Personal Computers	Home/Enterprise	Local home/enterprise contexts	C-S or Peer-to-peer
2000s	Centralized	Cloud Computing	Internet's users	Remote global context	Client-Server
2010s	Decentralized	Edge/Fog Computing	Every users	Local strong pervasive context	Cooperative
2020s	Hybrid	Post-Cloud Resource Continuity	Every users	Local pervasive and global context	Cooperative

Table 1: An overview of the characteristics of the different evolutionary ages in computing approach.

In this context, researcher embraced the idea of moving part of the computation from the Cloud to the *Edge*, i.e., closer to the data source. This means that edge devices, such as sensor equipped embedded devices, locally perform all or part of the pre-processing operations (e.g., aggregation, filtering. . .), so that only useful data are sent to the Cloud for the analysis and processing steps. This is a new turning point towards a revisited decentralized approach. This change has several benefits: (1) decreased network bandwidth requirements, and the probability of incurring in network congestion and other undesired side effects; (2) reduction of Cloud access costs; (3) contribution to reducing the growing CO2 impact of data-centres. Besides these benefits, some limitations are still relevant. First, Edge computing still relies on Cloud infrastructures and Internet connection availability. Second, there are no standard architectural models to integrate different Edge platform vendors and research technologies. Finally, there are still issues related to energy budget management and processing capabilities of edge devices. In particular, the energy budget (of batteries) and the thermal management represent the upper bound to the exploitation of further performance rooms.

The requirements of the emerging scenarios, together with the limitations of Cloud-based solutions, ask for an intermediate processing layer, that could fill part of the performance gap between Edge/IoT devices and Cloud computing servers. This novel concept, called *Fog Computing*, was proposed in 2012 by Cisco [6], and it has been translated into different paradigms [24], including Mobile Edge Computing [8], Dew Computing [21] and Mobile Cloud Computing [9, 15]. Fog Computing aims at building a scalable horizontal computing infrastructure closer to the IoT layer. This has the following advantages: (1) the possibility to complete computing-intensive tasks without the need to access a remote Cloud infrastructure; (2) achieving scalability at local level; (3) providing stronger guarantee in terms of security, by keeping sensible data at local level; (4) reducing the costs due to Cloud services and Internet connection subscriptions; (5) opening to real-time applications. In this regard, the current computational capabilities of modern embedded multicore platforms and mobile devices [22] make them good candidates for building Fog Computing layers.

Of course, there are some pending issues at different levels. First, despite some attempts, standard and homogeneous architectural models needed to integrate Fog and Edge computing are still missing or based on proprietary solutions. Moreover, exploiting mobile and other energy-constrained devices requires the monitoring of their availability. Overall, this means that a certain degree of reliability of the platform is needed, in order to guarantee the continuity of the service provision. Finally, regarding the heterogeneity of the devices, current solutions implement in part (or not at all) fine-grained resource management strategies. This means taking into account the device specific capabilities, in order to balance the workload placement and optimize the resources utilization, according to task-level performance requirements and energy efficiency maximization objectives [23].

Many-in-One: Resources Continuity. In this emerging research context, our goal is twofold: (1) Rearranging and integrating different approaches in a single high-level solution, based on a fully-distributed, cooperative and multi-layered infrastructure. (2) Proposing a solution to manage workloads at run-time, with performance and reliability guarantees, in a resources continuity perspective [12], through an analysis of open research challenges. In this sense, we aim at suggesting a new hybrid approach to bind together both centralized and decentralized post-cloud solutions.

The rest of this paper is structured as follows. The main contribution about the architectural model and the resource management strategy is presented and detailed in Section 2. Section 3 analyzes a smart-farming use-case and the potential benefits introduced by our approach. Section 4 concludes the paper exposing the current challenges and future research directions.

2 MANAGING RESOURCES CONTINUITY

To support the emerging use-cases, we propose a cooperative approach that provides a dynamic, modular and heterogeneous distributed system. This gathers the different post-cloud paradigms by implementing a resource continuity from the most remote platforms to the nearest-to-data devices, as shown in Figure 2. Suitable management layers must take into account performance requirements, QoS constraints and energy efficiency, going beyond the existing platform supports. In general, such infrastructure can offer flexibility and energy efficiency computation opportunities – an important challenge in the HiPeac Vision – as well as *vertical isolation* capability for real-time applications. However, the availability of links and devices may be subject to variability, due to mobility, energy budget, thermal stress, faults or other environmental conditions. In such a complex scenario effective management approaches are needed.

2.1 Two-dimension resource space

As highlighted in Figure 2, the infrastructure is characterized by a multi-tier architecture, in which *computing nodes* at different levels expose heterogeneous capabilities (in terms of processing, storage and network connectivity). A resource management approach should therefore operate on a bidimensional space.

By moving along the *vertical dimension* we can cross the different paradigms. At the top, Cloud computing platforms offer remote,

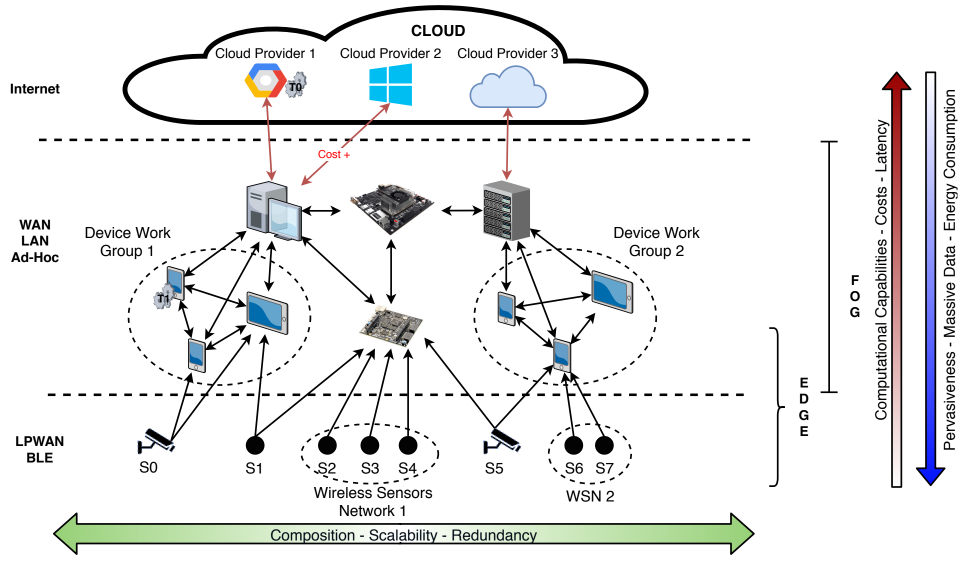


Figure 2: The layered architecture for post-cloud computing solutions.

reliable and powerful computational resources (i.e., PaaS[1], IaaS[2], HPC solutions). At the bottom, the Edge computing level, with low-power embedded devices and sensor nodes, for collecting data and performing minimal processing. In the middle we find the Fog level, with devices spanning from desktops to micro-servers, from gateways to high-end embedded p (e.g., NVIDIA Jetson TX2, Raspberry Pi, HiKey 960 …) and mobile devices (e.g., smartphones, tablet …). Going upward from the Edge to the Cloud level, we gain performance rooms and energy budget (we go from battery-supplied devices to plugged platforms). However, costs typically increase (due to computation infrastructure and connectivity), the dependency on a internet-based connection is stronger (e.g., for accessing the Cloud) and higher network latency are often experienced. To the contrary, moving from the Cloud to the Edge, leads to more pervasiveness and low-power consumption. Moreover, the data-to-computing distance is lower, which means lower latencies. However, the generated data require further aggregation and refinement, before they could be sent upwards for processing.

The *horizontal dimension* instead, allows to scale and balance the workload among sibling nodes. In particular: (1) at Cloud level, we can switch between different providers, to find the cheapest or most suitable solution, taking into account the computing center location; (2) at Fog level, we can compose different and dynamic worker groups (or clusters [4]) to cope with energy-efficiency workload allocation and privacy concerns; (3) at Edge level, sensors and Edge device can be composed together. Depending on the amount of generated data and load demand, we can exploit close-by nodes, to scale performance while maintaining acceptable communication latency. In case of heterogeneous nodes/devices, it enables the possibility of exploring different workload distributions, to find the most energy-efficient solutions. Moreover, the redundancy of devices and sensor nodes allows us to implement fault-tolerance policies, thus gaining in terms of reliability. On the other side, the presence of mobile devices can increase the complexity of the management strategies, due to the probability of unavailability.

Overall, this two-dimension space is strongly characterized by *heterogeneity*, in terms of both connectivity and computing resources.

Connectivity heterogeneity. Communications intra- and inter-level rely on different connectivity technologies. Cloud computing infrastructures are accessed via Internet-based connections, (optical fiber, ADSL, 3G/4G mobile networks, …), requiring the subscription of a service contract. At Fog level, devices can be grouped in different subnets, leveraging both wired and wireless solutions of different size and extension (e.g., Ethernet, WiFi, 5G). For instance, ad-hoc wireless/wired networks are useful to create on-demand work groups among temporary available devices (e.g., in automotive and traffic scenarios, public areas …). Edge devices for example, can be organized into Wireless Sensors Networks (WSN). Among the solutions emerged in the latest years, we consider the Low Power Wide Area Networks (LPWAN) (e.g., LoRafor long-range connectivity and Personal Area Networks (PAN) solutions, like ZigBee , or Bluetooth Low Energy (BLE) , for short/medium-range connections.

Resource heterogeneity. We can consider three types of resource heterogeneity: (1) *Inter-level heterogeneity*: computing resources with very different capabilities depending on the level (2) *Intra-level heterogeneity*: different computing resources, located at the same level. (3) *Intra-node heterogeneity*: heterogeneous resources available on the same device/node (e.g., general-purpose single or multi-core CPUs, many-core GPU, HW accelerators).

[1]Platform-as-a-Service
[2]Infrastructure-as-a-Service

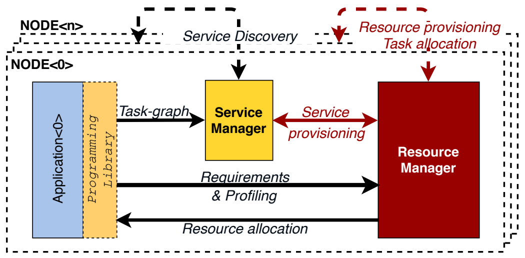


Figure 3: Resource and service-aware programming model.

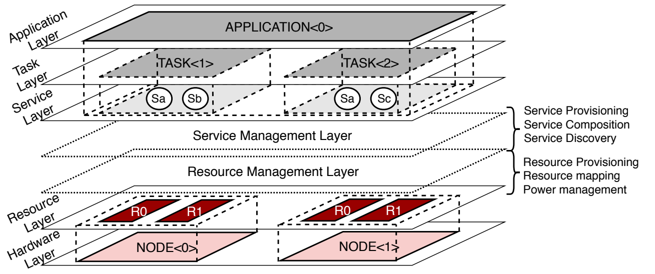


Figure 4: Multi-layer approach to address task-based applications, services and resources management.

2.2 Modular applications

The exploitation of such a distributed and heterogeneous infrastructure requires the adoption of suitable programming models, enabling a decomposition of the application into multiple *tasks* and *services*. A task can be considered an execution unit, that we should be able to deploy on different devices/nodes, featuring heterogeneous architectures, and capable of exchanging data with other tasks. A service, instead, is a functionality provided by a platform. Available services can be used to map specific tasks [10].

In HPC, *Message Passing Interface (MPI)* and *OSGi* are well established programming approaches used to develop distributed applications. *OpenCL*, instead, allows us to develop massively parallel applications, dealing with the availability of heterogeneous set of computing units on the current node. We consider a combination of these programming models as a possible starting point. However, to pursue the objectives previously discussed, a strong integration between programming model and run-time management system is required. In [1] an example of integration for single-node heterogeneous systems is discussed.

For distributed heterogeneous systems integrating layers from Edge to Cloud, we think that a suitable novel programming model is required. In Figure 3, we provide an overall view of desirable features, which are:

- An API to enable the exchange of data (profiling, task requirements, variation of allocated resources, etc…) between the application and the run-time manager;
- An interface to specify per-task application requirements (time constraints, QoS, reliability level, etc…);
- A managed execution model through which seamlessly reconfigure or migrate tasks onto different devices, according to performance, energy or reliability improvements opportunities;
- Compatibility with service-oriented architectures, exposing an interface to define task-service requirements and dependencies.

2.3 Multi-layer management

Once we have modular applications that we can distribute over the Edge/Fog/Cloud integrated infrastructure, we need a run-time management system in charge of establishing how to deploy the application in terms of task placement and service binding. To this end, on top of the aforementioned resource space, we can think of a multi-layer management approach, as shown in Figure 4. This must: (1) provide a well-structured and flexible stack, with homogeneous interfaces between layers, to deal with both service and non-service-based applications; (2) allow the identification of layer-specific requirements and their assessment, being agnostic with respect to the other layers.

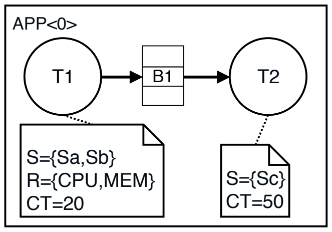
In particular, in case of a service-oriented architecture, since matching each task's service requirements and nodes' offer is a tricky aspect, that has to be addressed specifically, we can think of a dedicated *Service Management Layer*. This must be characterized by (1) an efficient mechanism for node's service provision and discovery; (2) a service composition strategy. However, each service may need specific resource requirements and constraints to be considered, in order to optimize the utilization of the service itself. In this case, the key-point is the presence of a novel *Resource Management Layer*, able to map the top-most layers to the assignment of the underlying resources. This must rely on the underlying *Resource layer* that provides the abstract resource continuity view over the infrastructure, where the *Hardware (or Node) layer* physically constitutes the aforementioned two-dimension resource space.

2.4 Run-time management in action

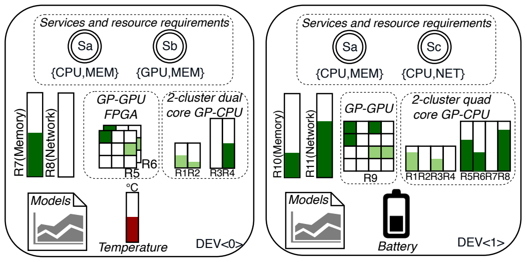
As argued above, from a system-side perspective, energy-efficiency and reliability must be maximized, while minimizing the costs. From an application perspective, the per-task requirements must be satisfied by taking into account the inter-task dependencies, the target architectures and the overhead occurring while transferring data among nodes/devices.

Figure 5 shows the proposed management flow of our infrastructure. The application work-flow can be represented through a task-graph, like shown in Figure 5a. For each task, we can specify performance requirements, time constraints and explicit requests of resources and services. In the example of Figure 5a, task $T1$ requires two services (Sa and Sb), an explicit set of resources (CPU and MEM), and specifies a time constraint ($CompletionTime$ = 20ms). In case of more accurate profiling, the application could come with data about energy consumption, for each possible target resource and operating point [14].

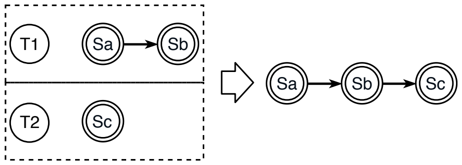
On the node/device side, we can expose information about available services (through the *Service layer*), the current status of resources (e.g., power consumption and temperature see Figure 5b). When the application starts, the composition manager can discover the available services and compose/match the *conversation*. This



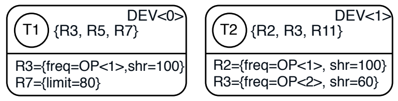
(a) The task-graph of a sample application composed of tasks $T1 - T2$ which write and read from buffer $B1$. Applications developer can explicit a set of requirements related to the tasks.



(b) The hardware equipment and status of nodes 0 and 1. The color inside resources shows the utilization level; different green brightness are related to the current core frequency.



(c) Left: Task-services dependency and conversations (e.g., task $T1$ specifies the conversation of services Sa and Sb. Right: the composed final conversation.



R3={freq=OP<1>,shr=100}
R7={limit=80}

R2={freq=OP<1>, shr=100}
R3={freq=OP<2>, shr=60}

(d) Node level allocation and settings. E.g., $T1$ is mapped to $R3$ (core 1 of CPU's cluster 1), $R5$ (accelerator) and $R7$ (memory). $R3$ frequency is set to $OP1$ and his share quota is 100 while the utilization of $R7$ is limited to 70%.

Figure 5: The proposed work-flow for application, allocation of tasks and services and infrastructure resource management.

determines the order in which services are used. Figure 5c continues the example showing task-services dependency and associated conversation. At this point, the service manager builds the final conversation. Different description specifications and composition strategies can now be implemented, like the promising ontological model, which offers a higher-level of abstraction and expressiveness, by adding context-aware properties to standard semantic web services [19, 20].

The next step involves the resource management layer in two ways: (1) global assignment of tasks across the nodes. (2) local resource allocation and power management (Figure 5d). In other words, a *global* resource manager instance will be in charge of dispatching the tasks over the nodes, considering also the service conversation provided by the service manager, i.e. only the nodes that exposes the selected services. A *local* resource manager that acts at level of node or group of devices, will be responsible of the resource reservation (CPU time, cores, memory) and the enforcement of power management actions (e.g., frequency-voltage operating point selection), by taking into account the characteristics and the requirements of the dispatched tasks.

Overall, global and local resource manager instances must cooperate. They do so by relying on (1) multi-objective *policies*, capable of exploring large multi-dimensional decision space, according to the aforementioned requirements and objectives; (2) run-time monitoring processes, to exchange information about the current status of nodes and devices, and trigger resource allocation changes when needed.

3 USE CASE: SMART FARMING

In order to highlight the possible use and the benefits of our infrastructure, in this section we discuss a possible use case related to a Smart Farming scenario. Let us consider an extensive plantation, with a smart irrigation system, based on information about the state of the plants (e.g. water need, diseases). This information is obtained through a WSN and a set of drones overflying the area. Moreover, IoT enabled tractors help farmers to plow the land in an efficient way. In this scenario, actuators can offer a solution to problems like dryness and diseases, by controlling irrigation path and dispense pesticides only to a specific part of the plantation. In addition to this, a Cloud-based weather forecast system would allow proactive response plans. In such a scenario, we can operate at different levels by exploiting all the actors involved: (1) *WSN nodes* that perform pre-processing of raw data, and send the intermediate output to more powerful nodes. Being battery and solar-powered, a resource manager could bound the exploitable computing capabilities, according to forecast and current charging status. (2) *Drones* that perform image pre-processing, in order to identify possible disease. Moreover, they can interact directly with nearby sensors to collect data related to specific plants or areas. (3) Different *Fog nodes* – i.e., fixed node, IoT tractors, farmers' smartphones – spread over the plantation, could receive aggregated data or computing task from (1) and (2). In this way, they could perform further analysis or compute fast emergency response. (4) *IoT* tractors could exchange information with nearby sensors, in order to adapt their plowing plan on the basis of the field conditions. (5) *Smartphones* could collect information from nearby sensors and drones to provide real-time information to the farmer. (6) *Cloud-based software* could collect all the aggregated information from the field to provide an overview of the plantation, with a further possible data analysis. Moreover, they could build area-specific weather model, to drive water and irrigation management plan, as well as to guide the grow and plowing schedule. (7) Finally, Fog controls *actuators* on the basis of information received from the Cloud infrastructure and other nodes. We believe that in this scenario the proposed resource management strategy could play a key role, i.e. optimizing the allocation and execution of the different tasks according to environment status, actual amount of processing to perform and current status of all the devices that my be involved.

4 CONCLUSIONS AND CHALLENGES

In this paper we discussed a run-time resource management approach for a distributed and heterogeneous infrastructure, able to leverage the resource continuity from Edge nodes to Cloud platforms. To effectively develop and adopt this infrastructure, which serves as a general model for subsequent research works, we need to face different open challenges.

Heterogeneity and Interoperability. Dealing with heterogeneous and distributed systems means relying on homogeneous interfaces. In this sense, it is possible to extend the open-source BarbequeRTRM [2, 3] framework by providing abstraction layers on top of different architectures [13] (e.g., multi-many core, embedded and mobile) and implementing global and local resource management strategies [17]. However, collecting information about the actual availability of remote and mobile devices and not just of connection type, is still lacking.

Application Modularity and Reconfigurability. Developers need a multi-task programming model that makes them agnostic with respect to the hardware target, and that allows them to specify per-task constraints and requirements. Extending already proposed programming models, in order to integrate the resource management layer, is another key activity.

Privacy and Security. Integrating Edge and Fog can introduce benefits also in terms of security and privacy, that a centralized Cloud-based could not guarantee [11]. Close-to-data processing reduces the volume of data transfers over the network, thus diminishing the risk of attacks. Moreover, users can define their own security policies (e.g., trusted devices...), to drive the task allocation, also addressing privacy issues, due to data processing on public devices, which can be hijacked by traditional attacks [18]. For this, isolation capabilities and cryptography algorithms are required.

System Reliability. Ensuring the reliability of the system is a key challenge for the execution of real-time applications. There are four areas of interest, spacing from hardware, software and data of single nodes to the global network [7]. The complexity of the system requires a scalable fault-tolerant mechanisms, able to provide multiple reference data sources and computing units to migrate the tasks. With respect to this, status information about nodes/devices [5] can feed machine learning algorithms in order to build more accurate reliability models.

5 ACKNOWLEDGMENTS

This work has been partially funded by H2020 RECIPE project grant 801137 and M^{2}DC project grant 688201.

REFERENCES

[1] G. Agosta, W. Fornaciari, G. Massari, A. Pupykina, F. Reghenzani, and M. Zanella. 2018. Managing Heterogeneous Resources in HPC Systems. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM '18).* ACM, New York, NY, USA, 7–12. https://doi.org/10.1145/3183767.3183769

[2] P. Bellasi, G. Massari, and W. Fornaciari. 2012. A RTRM proposal for multi/many-core platforms and reconfigurable applications. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on.* IEEE, 1–8. https://doi.org/10.1109/ReCoSoC.2012.6322885

[3] P. Bellasi, G. Massari, and W. Fornaciari. 2015. Effective Runtime Resource Management Using Linux Control Groups with the BarbequeRTRM Framework. *ACM Trans. Embed. Comput. Syst.* 14, 2, Article 39 (March 2015), 17 pages. https://doi.org/10.1145/2658990

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6 (2009), 599 – 616.

[5] S. Corbetta, D. Zoni, and W. Fornaciari. 2012. A Temperature and Reliability Oriented Simulation Framework for Multi-core Architectures. In *2012 IEEE Computer Society Annual Symposium on VLSI.* 51–56. https://doi.org/10.1109/ISVLSI.2012.22

[6] F. Bonomi et al. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12).* ACM, New York, NY, USA, 13–16.

[7] H. Madsen et al. 2013. Reliability in the utility computing era: Towards reliable Fog computing. In *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP).* 43–46. https://doi.org/10.1109/IWSSIP.2013.6623445

[8] ETSI. 2015. Mobile-edge computing-Introductory technical white paper. Retrieved May 10, 2018.

[9] D. Huang and H. Wu. 2017. *Mobile Cloud Computing: Foundations and Service Models.* Morgan Kaufmann.

[10] S. Kalasapur, M. Kumar, and B. A. Shirazi. 2007. Dynamic Service Composition in Pervasive Computing. *IEEE Transactions on Parallel and Distributed Systems* 18, 7 (July 2007), 907–918. https://doi.org/10.1109/TPDS.2007.1039

[11] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani. 2017. A Lightweight Privacy-Preserving Data Aggregation Scheme for Fog Computing-Enhanced IoT. *IEEE Access* 5 (2017), 3302–3312. https://doi.org/10.1109/ACCESS.2017.2677520

[12] X. Masip-Bruin, E. Marin-Tordera, A. Jukan, and G-J. Ren. 2018. Managing resources continuity from the edge to the cloud: Architecture and performance. *Future Generation Computer Systems* 79 (2018), 777 – 785.

[13] G. Massari, C. Caffarri, P. Bellasi, and W. Fornaciari. 2014. Extending a Run-time Resource Management Framework to Support OpenCL and Heterogeneous Systems. In *Proceedings of PARMA-DITAM '14.* ACM, New York, NY, USA, Article 21, 6 pages. https://doi.org/10.1145/2556863.2556868

[14] G. Massari, F. Terraneo, M. Zanella, and D. Zoni. 2018. Towards Fine-Grained DVFS in Embedded Multi-core CPUs. In *Architecture of Computing Systems – ARCS 2018*, Mladen Berekovic, Rainer Buchty, Heiko Hamann, Dirk Koch, and Thilo Pionteck (Eds.). Springer International Publishing, Cham, 239–251.

[15] G. Massari, M. Zanella, and W. Fornaciari. 2016. Towards Distributed Mobile Computing. In *2016 Mobile System Technologies Workshop (MST).* 29–35. https://doi.org/10.1109/MST.2016.13

[16] C. Puliafito, E. Mingozzi, and G. Anastasi. 2017. Fog Computing for the Internet of Mobile Things: Issues and Challenges. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP).* 1–6. https://doi.org/10.1109/SMARTCOMP.2017.7947010

[17] A. Sansottera, D. Zoni, P. Cremonesi, and W. Fornaciari. 2012. Consolidation of multi-tier workloads with performance and reliability constraints. In *2012 International Conference on High Performance Computing Simulation (HPCS).* 74–83. https://doi.org/10.1109/HPCSim.2012.6266893

[18] I. Stojmenovic, S. Wen, X. Huang, and H. Luan. 2015. An overview of Fog computing and its security issues. *Concurrency and Computation: Practice and Experience* 28, 10 (4 2015), 2991–3005. https://doi.org/10.1002/cpe.3485

[19] A. Urbieta, A. González-Beltrán, S. BenÂăMokhtar, J. Parra, L. Capra, M. A. Hossain, A. Alelaiwi, and J. I. Vázquez. 2015. Hybrid service matchmaking in ambient assisted living environments based on context-aware service modeling. *Cluster Computing* 18, 3 (01 Sep 2015), 1171–1188. https://doi.org/10.1007/s10586-015-0469-1

[20] A. Urbieta, A. GonzÃąlez-BeltrÃąn, S. Ben Mokhtar, M. Anwar Hossain, and L. Capra. 2017. Adaptive and context-aware service composition for IoT-based smart cities. *Future Generation Computer Systems* 76 (2017), 262 – 274. https://doi.org/10.1016/j.future.2016.12.038

[21] Y. Wang. 2015. Cloud-dew architecture. *International Journal of Cloud Computing* 4, 3 (2015), 199–210.

[22] Y. Wang, I. Chen, and D. Wang. 2015. A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges. *Wireless Personal Communications* 80, 4 (2015), 1607–1623. https://doi.org/10.1007/s11277-014-2102-7

[23] M. Zanella, G. Massari, and W. Fornaciari. 2018. Enabling Run-Time Managed Distributed Mobile Computing. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM '18).* ACM, New York, NY, USA, 39–44. https://doi.org/10.1145/3183767.3183778

[24] Y. Zhou, D. Zhang, and N. Xiong. 2017. Post-cloud computing paradigms: a survey and comparison. *Tsinghua Science and Technology* 22, 6 (December 2017).