Living on the Edge Dependably: New Challenges and Solution Directions

Saurabh Bagchi (ECE, CS, Purdue University); Jitesh Panchal (ME, Purdue University)

Invited for NSF Workshop on Grand Challenges in Computer Systems Research, Mar 24–25, 2018

Overview

Edge computing is the practice of placing computing resources at the edges of the Internet in close proximity to devices and information sources. This, much like a cache on a CPU, increases bandwidth and reduces latency for applications but at a potential cost of dependability and capacity. This is because these edge devices are often not as well maintained, dependable, powerful, or robust as centralized server-class cloud resources. Till date much of the attention has focused on ways to architect applications to execute efficiently on the platform comprising client devices, edge devices, and cloud server farms. Now as this vision becomes compelling enough that important applications are moved to such a model, it is important to pay attention to the salient dependability challenges brought about by edge computing.

We drive this discussion by the example of automatic recognition of parts during product assembly on a shop floor as part of a drive toward cyber manufacturing. Consider that during product assembly through a mix of humans and robots, various parts are coming to the assembly line—there is some diversity in the parts as well as their quality. It needs to be determined what the part is and what the quality is before it is processed. Obviously, to maintain a high throughput of the manufacturing process, the above two tasks need to be done with low latency. Hence, there has been a move toward the use of edge computing in advanced manufacturing where some processing is done locally at the AR/VR device worn by the human operator, some at a local desktop-class machine on the shop floor, and if need be, expensive image analysis for quality determination at an off-premise cloud. We will show through a simple application of recognition of machine parts that current state-of-practice cannot handle failures, hard and soft, in the edge layer. The application runs collaboratively on a mobile device (simple recognition tasl), an edge layer of multiple laptops, and a cloud instance (the latter two are invoked for complex quality determination algorithms).

What is Different?

While several dependability challenges in the above model are shared with traditional cloud computing (fast error detection and failover, protecting against performance interference from co-located processing units, and automatic placement of computation to the best host), there are four challenges that are somewhat salient. *First*, the degree of over-provisioning of edge resources that is economically viable is much less than in cloud platforms. Thus, failover must be done to within a limited set of devices and then failover may be needed to a cloud server farm. This has to be done while meeting the low latency requirements that are fundamental in edge applications. *Second*, devices in the edge layer are often not continuously available. This happens because they are typically less well-managed, used opportunistically, and used in challenging wireless environments leading to fluctuating connectivity. Thus, dependability mechanisms have to be built with uncertain availability of individual devices or clusters of devices (such as, for wireless connectivity degradation) in mind. *Third*, it is not practically conceivable to require users to authenticate the client devices to the edge frequently. This is despite the fact that there will be some degree of churn in the client devices that are associated with any set of edge devices. *Fourth*, the issue of multi-tenancy on edge devices poses challenges because of the variety of capabilities and execution platforms that exist. In the cloud environment, hypervisors and virtual machines, and more recently, containers, have become standard part of the infrastructure. For edge platforms, this is not the case now and is unlikely to be the case in the future because of lack of standardization of the system software and some edge devices being too resource constrained to support VMs or even containers. Even if support does exist, there will be sharp differences in the number of VMs or containers that can be supported on each node.

What are Broad Solution Directions?

The four broad challenges can be solved through a combination of fundamental systems research, application domain-driven research, and engineering adaptation of current solutions. *First*, fast failover mechanisms need to be designed, also considering that overload conditions of edge devices may be frequent. *Second*, dependability modeling and design under uncertain availability (such as, for opportunistic mobile networks) need to be updated to handle the current challenge. Priors about patterns of availability of specific classes of edge devices can aid in increasing overall dependability. *Third*, macro authentication protocols need to be designed whereby authentication on a single device (say, on one client device using biometric means) can be used for authenticating on other devices at the same or different layers. Flexible notions of dynamic asset groups based on location

proximity would be useful in this context. *Fourth*, some standardization of containers for edge platforms and being able to trade off the performance isolation and the size of the container footprint would be relevant.

Pillars to Build Upon?

The proposed solution directions build upon some foundational pieces that have already been researched and developed. Broadly, they fall under four topics. For space reasons, we confine ourselves to giving a small number of the most relevant references and do not expand upon them.

1. *Dependability in distributed applications*: including error detection [1], diagnosis [2], and recovery (such as, failover) [3] through distributed protocols.
2. *Mobile offload*: which encompasses decisions on whether to execute on a mobile client or offload to a cloud platfirm, and includes application partitioning [4], dynamic decision making [5], and mathematical modeling of costs and benefits [6].
3. *Macro operations*: including macro programming [7] and macro reconfiguration [8] of multiple devices (popularly wireless embedded nodes or sensor nodes) with identical or closely related program or configuration requirements.

References

[1] G. Bronevetsky, I. Laguna, B. R. de Supinski, and S. Bagchi, "Automatic fault characterization via abnormality-enhanced classification," in *42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–12, IEEE, 2012.

[2] G. Khanna, M. Y. Cheng, P. Varadharajan, S. Bagchi, M. P. Correia, and P. J. Veríssimo, "Automated rule-based diagnosis through a distributed monitor system," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 266–279, 2007.

[3] R. Strom and S. Yemini, "Optimistic recovery in distributed systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 3, no. 3, pp. 204–226, 1985.

[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth European conference on Computer systems (Eurosys)*, pp. 301–314, ACM, 2011.

[5] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services (Mobisys)*, pp. 49–62, ACM, 2010.

[6] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[7] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using kairos," in *International Conference on Distributed Computing in Sensor Systems*, pp. 126–140, Springer, 2005.

[8] L. Mottola, G. P. Picco, and A. A. Sheikh, "Figaro: Fine-grained software reconfiguration for wireless sensor networks," in *Wireless Sensor Networks*, pp. 286–304, Springer, 2008.