Key Requirements for Real Time Apps

ZEDEDA

Introduction

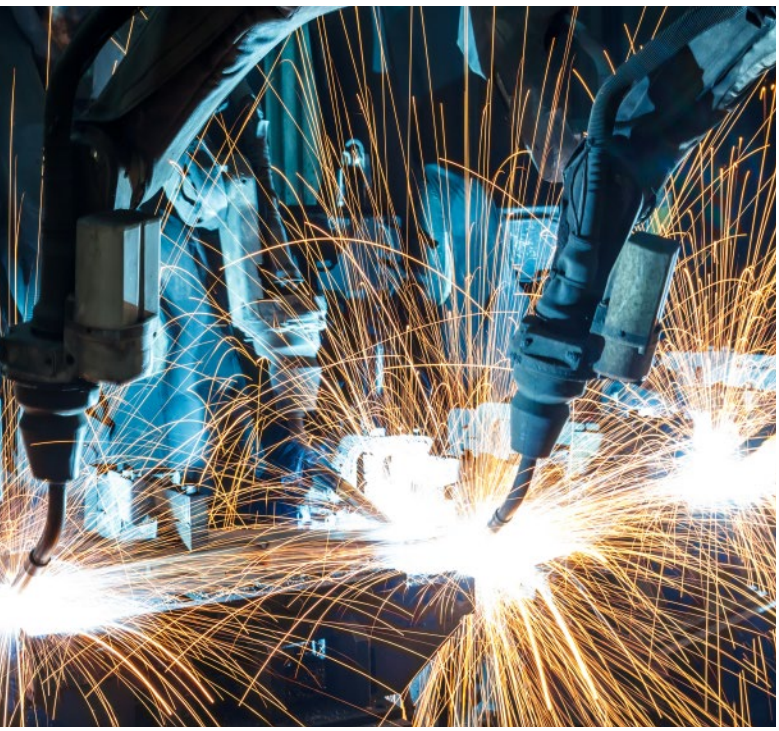
The IoT phenomenon is taking the world by storm but the impact of IoT is only now coming to light. With computing power getting less and less expensive, IoT is effectively digitizing every aspect of the physical world. Every interaction is generating data and providing a means for applications to take action. But the true power of this new cyber-physical world comes through when applications use data and take action in real-time.

Real-time apps are the backbone of digital transformation and real-time business. But for businesses to truly transform they can't rely on cloud technologies just moving to embedded systems and embedded systems can't support the agility and scale required to compete in a cloud world. Edge computing needs to be real-time AND cloud native. This paper outlines the essentials to start moving the industry toward a cloud native future.

What does it take to create "Real-Time Apps" and become a "Real-Time Business"?

Edge Computing is the route to the ultimate real-time business - because it eliminates latency, bandwidth, and autonomy limitations of cloud that can hamper or destroy real-time apps . However, edge computing as it stands today has never contemplated a connected, "hyperscale" world. Unlike cloud computing, edge computing is still very rooted in the embedded computing/processing world which wasn't intended to achieve app lifecycle management at "hyperscale" as the cloud is architected. The embedded world is less accustomed to networked systems, security, and app management that solve the problems of deploying, managing, and securing apps at hyperscale.

Of course, apps do exist today in the embedded computing world that are autonomous, real-time, and interact with



the physical world. For example, programmable logic controllers (or PLCs) on an assembly line controlling robots or in a solar farm controlling power systems. However, these embedded systems have operating limitations that, when viewed from the scale, security, and network requirements of an IoT world, make it hard to create a consistent approach to deploying, securing, updating, and future-proofing them, especially over-the-air at remote locations, without an unrealistic increase in operational costs.

The era of Cloud Computing introduced app developers to the concept of "cloud native[1]" apps – a model where developers are more concerned with how apps are created, deployed, and maintained at hyper-scale and not concerned with "where" they are physically deployed. App developers can focus on continuous, agile delivery of software with a base assumption that infrastructure should be taken for granted. "Cloud native" at the edge extends this base assumption to enable agile development regardless of the network topology, geography, or hardware diversity found at the edge.

> *Our recent research shows that while OT teams have the application plans for leveraging IoT, the vast majority of organizations' IT resources and capabilities are maxed out. This leaves open the question of how these edge applications and IoT will scale out without compromising security or taxing resources even further in the future.*
>
> *Christian Renaud*
> *Research Director, IoT at 451 Research*

1 - Pivotal, Inc. - https://pivotal.io/cloud native

Examining software app lifecycle in the edge today versus a "Cloud Native" approach side-by-side highlights the fundamental differences that an edge app platform must achieve to provide continuous delivery at distributed location and at hyperscale.

App Lifecycle	Embedded Software **Edge Today**	Cloud Native Approach **Edge Tomorrow**
Design	Hardware Dependent	Platform Independent
	Waterfall (gate based)	Agile (iterative)
	Monolithic	Micro-services & Containers
Develop	Reinvent wheel (eg many RT OS'es)	Re-use centric
	Legacy programming models (C, ASM)	Modern small footprint programming (Go, Node.JS)
	Proprietary, expensive dev tools & closed-source	Open source, open development
Deploy	"Burn-in" a the factory with testing prior to shipping	Instant app deployment when and where needed; Zero-touch full stack deployment
	Update in field as little as possible	As many times as needed at the moment it is available
	High risk for upgrades, one way. Manual Disaster Recovery	Roll-back ability Automated verification and disaster recovery
Operate	One at a time management	At scale management with "intended state" model
	Local focused (console/serial/shell)	Network/remote focused, SD-WAN overlay integrated with apps
	Insecure due to being air-gapped (not connected) historically	"Ground to Cloud" defense in layers with PKI cryptography;
Analyze	Not easy to get statistics/diags out of device	Continuous statistics, diags and analysis
	Distributed and "ships in the night"	Centralized and aggregated
	No feedback loop / lack of automated analytics	Machine learning and insight driven

This cloud native approach has allowed the largest cloud companies achieve operations at hyperscale and is an absolute requirement for IoT to live up to its much hyped $1.3 Trillion-dollar forecasts[2]. To provide a view of what "hyperscale" will look like in edge computing we need to look no further than the recent "Meltdown" bug discovered by Intel. To protect customers from effects of the "Meltdown" bug, Amazon Web Services could have had to quickly patch on the order of 2-3

2- IDC Worldwide Semiannual Internet of Things Spending Guide, Dec 2017

Millions servers to insure operations and security for all their customers (this is an estimate, its obviously a proprietary number they don't share). These fixes were not deployed in weeks or months but hours and days.

Why is this approach needed for the edge? Ford Motor Company shipped ~6.7M cars in 2016. Assuming that number stays constant, the modern car can have several servers (with multiple software containers or virtual machines), there is an inevitable future edge computing scenario where a single car company could have to deal with 10-times the number of upgrades required by a cloud company the size of Amazon to protect their customers! How does a car company scale its operations to deal with compute and app problems? Are they going to increase the IT team to match Amazon Web Services? These are realities of dealing with hyperscale at the edge!

Operations Technology needs Real-Time Apps, Real-Time Apps Need Cloud Native

"Cloud Native" is a computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self healing multi-tenant nodes[3]. The term is most often associated with developing cloud-based apps. Everything that goes into making the virtualized infra-structure of the cloud transparent to the app developer is what makes the environment "cloud native" The principle here that is critical to edge computing is creating an environment that's "capable of scaling to tens of thousands of self healing multi-tenant nodes". In the cloud as its used today, those "nodes" are in datacenters packed with infrastructure and connected with incredibly fast Ethernet networks. But to truly build apps that are "cloud native" and run at the edge, the "nodes" look very different from a datacenter. They are not always servers, always wired, or even always stationary! Cloud native at the edge has to provide the same developer experience that a cloud native datacenter (i.e. private or public cloud) would provide which means making the incredibly diverse, geographically distributed edge environment and automatically account for the new conditions without changing how developers work.

That is a daunting task, however here are some key considerations when looking to create a cloud native environment for edge apps so they can work with micro-second precision. But before diving into the key considerations, it is important to look at a map of vendors who play an important part of assembling the whole picture should an organization endeavor to build their our own solution.

3 - Cloud Native Computing Foundation, Mar 2017 - https://www.cncf.io/about/charter/

3 Key Requirements For Creating Cloud Native Environment at the Edge

With an ever-growing list of pieces to integrate to create a cloud native environment for real-time edge apps, it can be confusing. At a minimum, when assembling your edge technology stack, there are three main categories of consideration: Device Software/Security, Management, and Networking.
Device Software/Security refers to how the device's full stack operates – from hardware through user application. A key element is creating sufficient abstraction from the underlying IT infrastructure so that applications can request services simply and securely.

Management discusses key elements to implement to insure deployment and ongoing management can be done without significant increase in operational costs.

Networking is the most unique element of the edge environment. Whether apps are hosted on minute devices built into cars or in distributed co-location centers, the applications have to be abstracted out of absolute network assumptions and can just target locations, not complicated networking situations.

Device Software/Security

- Keep "Isolation" as the guiding principle: This is key to designing a system that is more reliable and secure. Implement a system that keeps the apps separate from device orchestration and operating software. Its one of the key tenants of creating a cloud native development approach. The goal is to create an edge that will support an agile development approach and this concept will allow the capabilities to change with business needs and also keep up with the latest security changes (key now that everything is connected to the Internet). Continuous deployment and improvement of your app code will be required.

- Ensure that single app failure doesn't affect base operating system. This is another element of isolation. If the apps are sufficiently isolated then a failure of an app doesn't translate to failure of the entire box. Why is that important? The answer is probably in your pocket. Imagine if every time an app on your phone failed or locked up the entire phone needed to be manually restarted…or had to have the battery removed. Now imagine doing that for 5000 ATM machines around the world. The goal is to add "apps" not "options".

- Keep the footprint of app and device management elements to a minimum. The edge computing devices, by definition, are typically in remote locations where there is limited network connectivity. Having to download the large files of drivers, unused apps, etc each time for these devices might mean long delay particularly when updating large scale deployments. More likely than not the applications are for businesses and extended periods of non-service directly impacts revenue – example: Fleet trucks and cars. The longer the maintenance window to do regular patches, the longer the loss of productivity. Efficient, lightweight apps with proper isolation is required.

- The Console Cable is NOT your friend! Eliminate any direct access to the device management software on the device - no local or remote console/shell access. This access can result in different configuration on the device that what's intended by the central management console. As large organizations try to scale out to a global deployment, edge devices will be in thousands if not millions. Access to the device software from anywhere other than the "master" console has 2 essential problems:

    1. When deploying devices at scale leaving local access can cause changes that the central authority didn't intend. What if the local admin changed the password? It would make security updates and global policy changes impossible leading to manual intervention (a.k.a. OPERATIONAL COST). No Shell or Local access, means no user name/passwords that allow access to the device.

    2. If there is local access or worse, remote console capability, this actually increases the security "attack surface", in other words, a way for nefarious actors to hack into the device either onsite or if the remote device is physically stolen.

  If the central management console communicates with the devices using APIs then a much richer set of standard security mechanisms can be used. No need for usernames and passwords on each device. Thus attacks like the Mirai botnet are designed out of existence.

  One last important point – the management console should track the intended configuration of every device in its role as the "master". If done properly, the simple requirement of "no console access" allows all policies to be enforced consistently over ten, ten thousand, or ten million devices.

Management for Geographically Distributed Devices

- Zero touch, full stack provisioning is the next objective when we talk distributed devices. Above we discussed a bit about maintaining thousands of devices, requiring no manual intervention, but how do you get them there in the first place? Map out the experience you want for any would-be provider of edge solutions – You want the remote site to just provide the power and connect the network cable – and maybe not even that, perhaps wireless if that complies with your security policies. Anything more and you can assume that there will be a truck roll and labor costs – not good when its done 10,000 times. Power and network cable and the edge device should then discover and retrieve its full stack – from OS through network and app provisioning - over secure channels.

- They are who we thought they were…Device identification – This is one of most the important considerations in your edge infrastructure. How to you identify each device? Each device might be running multiple applications. Each application may have its own structure about how they are connected to other resources, and other apps. Lets take a look at the scale of edge computing for a moment. In the scenario mentioned earlier, Ford shipped 6.7 MILLION cars in 2016 – they're a public company, that's easy to figure out. Each car has multiple embedded devices. That's on the order of 18 MILLION devices running apps that may need managing. For reference, Amazon Web Services has approximately 2-3 million servers that form their cloud.
  So there will be HUGE numbers of edge devices and very complex situations to navigate even in non-technology organizations. To make things even more complex in the edge as you consider device (and app) identification, consider also that the target device may be mobile (example: fleet of self-driving trucks)! The network address might change but the device needs to still provide the same services to its end user.
  Network reachability needs to be flexible, device identification needs to be rigid.

- Monitor everything! Or more importantly, implement everything so you can get data on how its all going at the system health level. Of course, every distributed system should be built with metrics, events and extensive logging and this holds true for edge computing infrastructure. The key considerations here are the sheer scale of edge and limited network connectivity. The system metrics collection should be smart, policy driven and should be bandwidth aware.

- The beneficiaries of IoT are almost never the IT department so make sure the system can be aware of the intended use and optimized for it. If the operator of the app or the assets need to sift through ten of thousand of devices and find the device and the isolated app on that device to act on, then you've failed. As users are typically in business operations, they don't need to understand intricate details of how edge computing stack works. Doctors can't spend time surfing a UI that is organized by server. Alice, as the fleet operator needs to know which Truck needs an upgrade and when they will be in shop, rather than her working through a list of devices needing upgrade and then mapping that to particular locations and coming up with an upgrade list based on that location map. Business context is key to the presenta-tion of data.

Networking for Real-Time Experiences

- Data drives apps, period. IoT is effectively digitizing every interaction that happens in the physical world and edge computing infrastructure allows us to collect data continuously. For apps to process and act in real-time that will be forced to the edge not just for latency reasons but also for bandwidth reasons (particularly wireless as telco infrastructures were not designed to massive upstream data over licensed bands). What a proper edge system should provide is the ability to take this stream of real-time data and provide access to multiple entities (example: 2 different types of analytics of the same video stream). Make sure to account for multi-tenancy of data which requires proper data-governance, data abstraction layers at edge node, and policy driven data access.

- Software Defined policies are key to a well-formed edge. The cloud still plays a critical role in overall digital transformation through IoT. Processed data at the edge still needs to be transported to the cloud for long term storage and batch processing. But the edge environment is complex, unpredictable, and in many cases mobile. Edge nodes need to intelligently understand and navigate the available network to transport critical, less real-time data to cloud. The software defined policies need to accommodate for available networks, location of device, data compression.

- App Permission and monitors: Edge will be everywhere – ubiquitous. There is no more "perimeter" when services that were typically centralized in the datacenter (Cloud) now have to move out to the edge and deal with traffic from neighboring systems, aka "east-west traffic". There is no perimeter, there is only data sovereignty. A properly designed system therefore cannot depend on perimeter based firewall. The edge devices themselves need to provide the app specific, user configurable (ACLs) with pass-word-less encryption mechanisms. Additionally, the edge device will need a way to have tools to con-tinuously monitor itself so it can determine if any app resident on the device is misbehaving or is under attack. Remember, the apps are isolated and separate from the device operation software. That device software should monitor and understand the difference between authorized and unauthorized behavior intelligently.

ZEDEDA.com | 2040 Martin Ave, Santa Clara, CA USA 95050