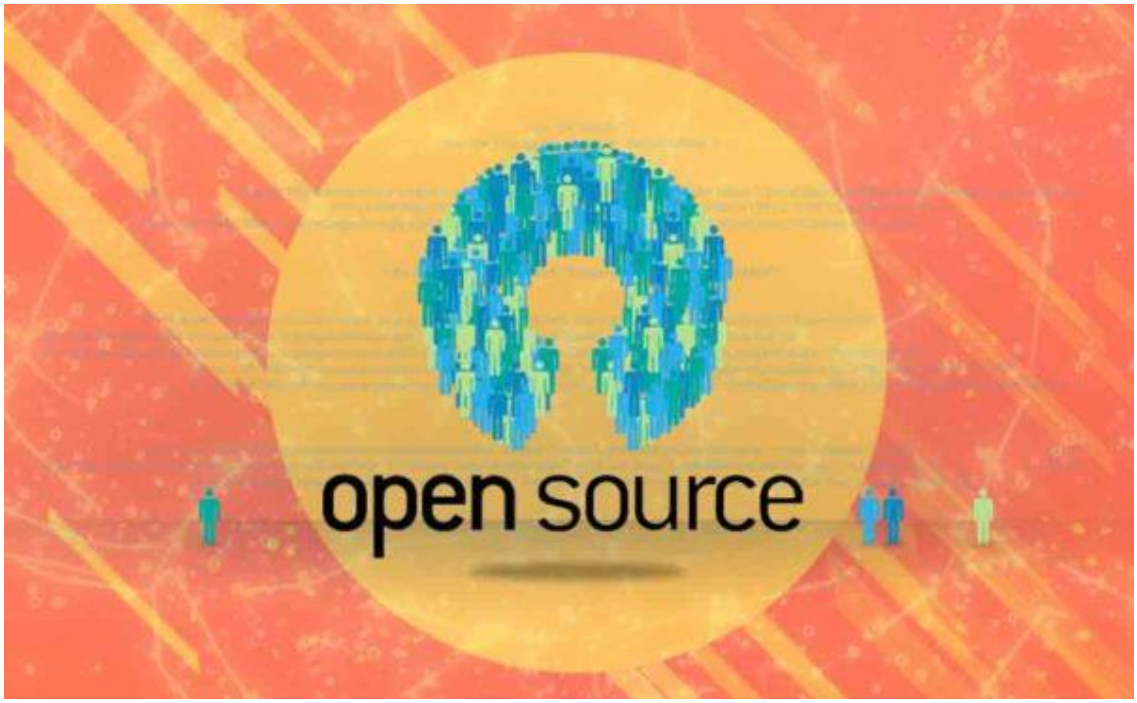
How to Choose Your IoT Platform – Should You Go Open-Source?



DataArt
October 10, 2017

In my previous article "Should you use Serverless Architecture for your IoT Solution?" I reviewed DataArt's approach to building a serverless IoT solution from scratch, its pros and cons, costs, and factors to consider before committing to this approach.

In this article, I'll present a more general overview of how to choose the right IoT platform for you. I will highlight the major criteria to consider and evaluate in a custom or open-source IoT platform. Finally, I'll present some math and cost analysis, comparing open-source to serverless IoT platforms, to help decide which approach to choose.

Cloud giants, including AWS, Azure, IBM and Google, make considerable investments into their cloud services, which makes them cost efficient, convenient, scalable, and fault tolerant. Custom IoT platform of choice should be no different.

As Head of IoT practice at DataArt, running DeviceHive, I've had to deal with a lot of challenges that all IoT platforms face. It helped me come up with the list of criteria to evaluate a production-grade IoT platform:

- Scalability
- Reliability
- Customization
- Operations
- Protocols
- Hardware-agnostic
- Cloud-agnostic
- Support
- Architecture & Technology Stack
- Security
- Cost

If you're familiar with these criteria, you can skip the next section and go directly to ***Analysis / Calculations***.

Scalability

When considering a serverless IoT platform, you should keep in mind that it scales out automatically to serve a growing load, without prompts for action. It supports business growth, preventing blackouts due to lack of resources behind the cloud.

While it is hidden behind the scenes with the serverless approach, a custom IoT solution must provide options to scale itself with predictable performance, memory and throughput metrics. Some IoT frameworks provide more advanced and fine-grained scaling options that are more efficient but may require a skilled team or advanced operations.

Reliability

Real-time scaling out to serve the changing load demand with more compute resources is not enough. In production, you will not have the luxury of determining the reasons for messages disappearing from devices or the whole platform not responding.

The system should be failover and have disaster recovery capabilities to support production levels of reliability. It should implement health checks, monitor its components and provide self-healing for most cases. When choosing your platform, pay attention to how its reliability can be achieved. It is usually a combination of architecture and operation-specific sets of factors.

Customization

Serverless IoT approach allows flexibility by providing integration options with the rest of its cloud services. However, in most cases, we can't modify the core of the IoT service itself. A custom IoT solution should provide great built-in functionality, extensive libraries, APIs and integration options with other platforms or services, an ability to extend the core of the system by using custom plugins or integrating custom code.

With open source solutions, we can go way beyond that. There is freedom of unlimited customization and tuning of the product to match specific needs. Open source also gives you the entire picture of its all ins and outs.

Operations

Operations are a significant, and often underestimated, factor. Unlike serverless IoT platforms, where all operations are mostly hidden under the hood, it can take tremendous effort for your team to operate a custom IoT platform. You should always consider platform orchestration capabilities, especially when the platform is built using the micro-services architecture.

Also, an important benefit is a dashboard displaying the overall system statistics and health information, status of each particular module or service, with an option for notifications in case of an emergency.

Protocols

It's worth considering in advance which protocols the custom IoT solution supports. If there is uncertainty about staying with a particular IoT platform in the future, I recommend checking whether it supports MQTT, the most broadly used protocol invented almost 20 years ago and a de-facto standard in IoT.

It provides an opportunity to almost seamlessly migrate to another IoT platform, by replacing its backbone – MQTT broker with another IoT platform which supports MQTT. Moreover, this protocol is widely adopted by most of the cloud IoT providers, including AWS, Azure, and IBM Watson.

Additional flexibility is gained if the IoT solution supports API over WebSockets, REST and CoAP.

Hardware-agnostic

The hardware manufacturing industry evolves extremely fast, and we can observe how small and powerful edge devices could be. Most of the devices today can run operating systems, making them extremely flexible and capable of running programs written in any language.

However, with low-end devices, one is constrained by their memory and computing power, and apps must be extremely efficient in resource consumption. Before choosing an IoT platform, it's worth checking which devices it supports, whether it has native implementations for low-end devices, or if it supports lightweight communication protocols (MQTT or CoAP) which can run on low-end devices.

Cloud-agnostic

In most cases, when considering a custom or open-source IoT platform, it's important to remember that its deployment, operations and maintenance are also part of the equation.

To make sure that the platform's flexibility is not limited, it should be deployable for operations in any infrastructure, be it cloud, on premises or a hybrid. If the platform heavily relies on particular cloud services, it will likely only work with this particular cloud. While it may be a good immediate decision to go with the cloud lock-in option, what if you decide to move to another cloud considering how quickly they evolve and how much they compete? You will end up changing the whole platform, which is a great waste of effort.

Support

Support is one of the services provided at extra cost by the companies that develop IoT platforms, ensuring that most recent updates, security issues and bugs fixed are delievered in a

timely manner. Support allows custom features to be developed and platform operated on one's behalf.

There are a few options for organizing platform support: hiring or educating a dedicated team, using support agreements with the company behind the platform, or relying on community support in case of an open-source platform.

A mixture of the above is probably the most effective, but it depends on a particular case. Even if no support is required anytime soon, this option should be investigated in advance, to avoid dealing with a non-working or not supported product. It also makes sense to explore how often new versions are rolled out, the product features road map, the number of staff members involved in support and development, and whether there are actual people to talk to.

This page contains a form, you can see it here.

Architecture & Technology Stack

The importance of architecture and technologies used in the platform is fundamental as its future maintainability depends on its original design.

The platform should employ production-grade, well-supported frameworks, tools, and languages, while limiting their variety in order to reduce the need for developers with respective skill sets.

Micro-services architecture should be considered among others. While it may be difficult to operate, it provides the most flexibility for substituting any service with a custom implementation, while keeping the rest of the system intact.

With an open-source product, you can do your personal due diligence to explore architecture and dive deep into the technology stack.

Security

Security in IoT is one of the hottest topics. The core problem is the absence of industry standards. While security concerns each component of the IoT ecosystem – devices, platforms, and applications – this article focuses on platform security.

When choosing an IoT platform, security must be one of the top priorities, built in from the very first phase of the project. Checking security features of an existing IoT platform is a must. At the very least it should provide a transport level security (TLS) to communicate with devices and applications, authentication for devices and users.

It is also worth researching whether the platform has authorization capabilities, what kind of data lifecycle management is used, whether it provides an option to store your data encrypted, and last but not least, if there are security audit documents available.

No one wants their business to become a botnet of compromised devices making DDoS attacks, or scanning website vulnerabilities. Recovering such 'infected' device-cloud-application

ecosystem and fixing security issues will cost a lot and take tremendous effort. Please don't let the security criteria slip off your list.

Cost

Different IoT system developers provide different pricing models. In most cases, an open source platform can be used for free. Below are the most common pricing models for IoT:

- pay per node/year
- pay per active device
- pay per message
- pay for premium features (optional)
- pay for support (optional)

The price often doesn't include the cost of the infrastructure to host the IoT solution, and it's usually the biggest part of the bill. I am going to break it down in the next section.

Analysis / Calculations

We will compare the costs of serverless IoT and open-source IoT platform usage in the cloud. For the sake of simplicity, we'll assume that an open-source IoT platform has the following attributes:

1. It scales linearly and predictably – each additional node can handle a constant number of messages per second;
2. Requests are evenly distributed over time with no spikes;
3. The platform operates in a guaranteed message delivery mode;
4. Message size is 1Kb;
5. One node instance can handle 5000 msgs/sec with reasonable latency less or equal to 200 ms.

First of all, let's compare the raw infrastructure costs. To make sure we compare apples to apples, we will analyze serverless AWS IoT and open-source IoT solution hosted on AWS infrastructure.

Our reference instance for open-source IoT solution will be c4.xlarge with 500 GB General Purpose SSD and 500 GB/mo backup space for each node. In addition, we'll count costs for one elastic IP address, one load balancer for cluster purposes and outbound traffic based on 1 Kb per message. For serverless IoT, we will consider a setup from my previous article and count only AWS IoT itself and DynamoDB services.

AWS IoT pricing model is based on received and sent messages. It also counts some service messages like acknowledgments, pings, and a few others, but we'll skip it for simplicity.

Let's assume that there is one device sending one message per second to the cloud. It will generate 2.628 million messages monthly, over 730 average hours a month.

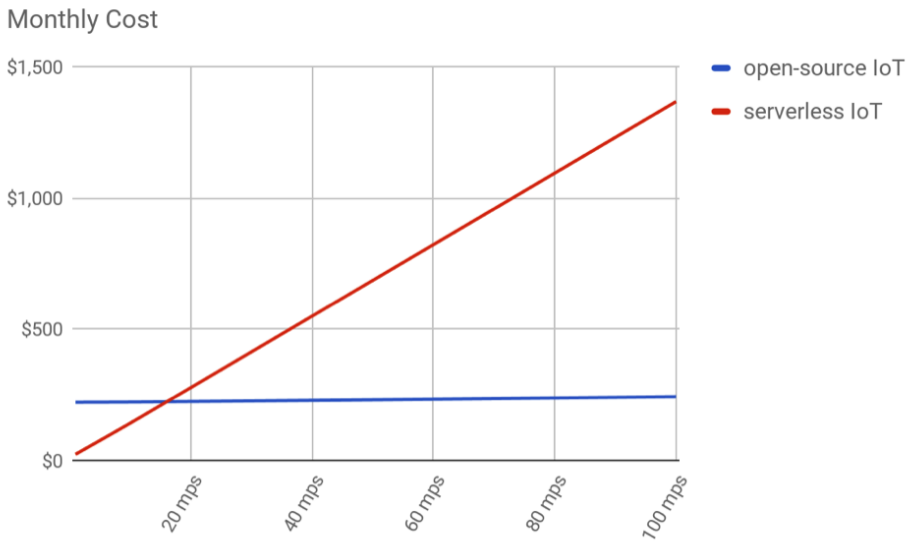
AWS IoT costs $5 per 1M requests, and DynamoDB costs $0.0065 for every ten put requests per second, and the same cost for every 50 get requests per second.

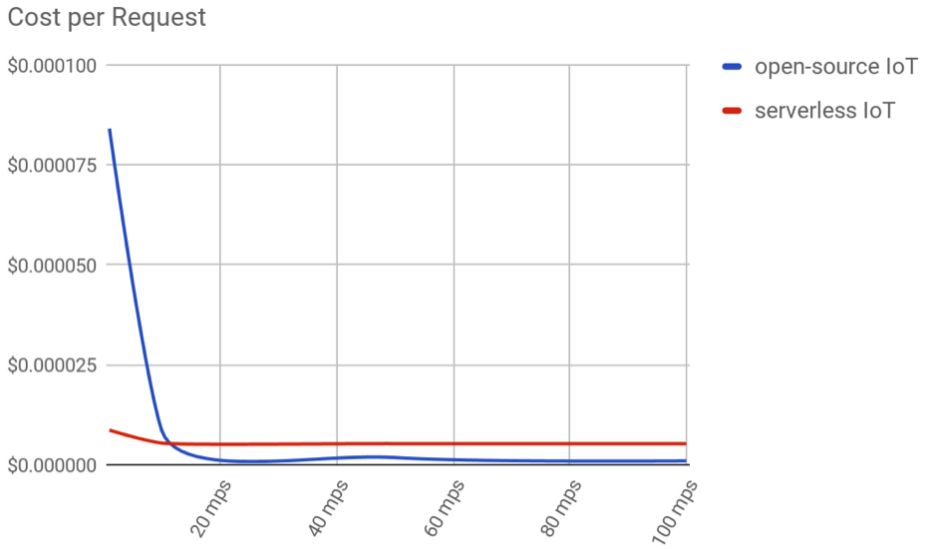
We see a very affordable monthly bill at around $22.

With open-source IoT solution hosted on AWS infrastructure, we will get a bill close to $225.

However, if we have ten devices sending 26.2 million messages monthly, we will get charged $140 for AWS IoT and still $225 for the open-source platform.

Let’s take a look at the below charts to see the monthly cost and cost per message correlation to a number of messages sent per second.





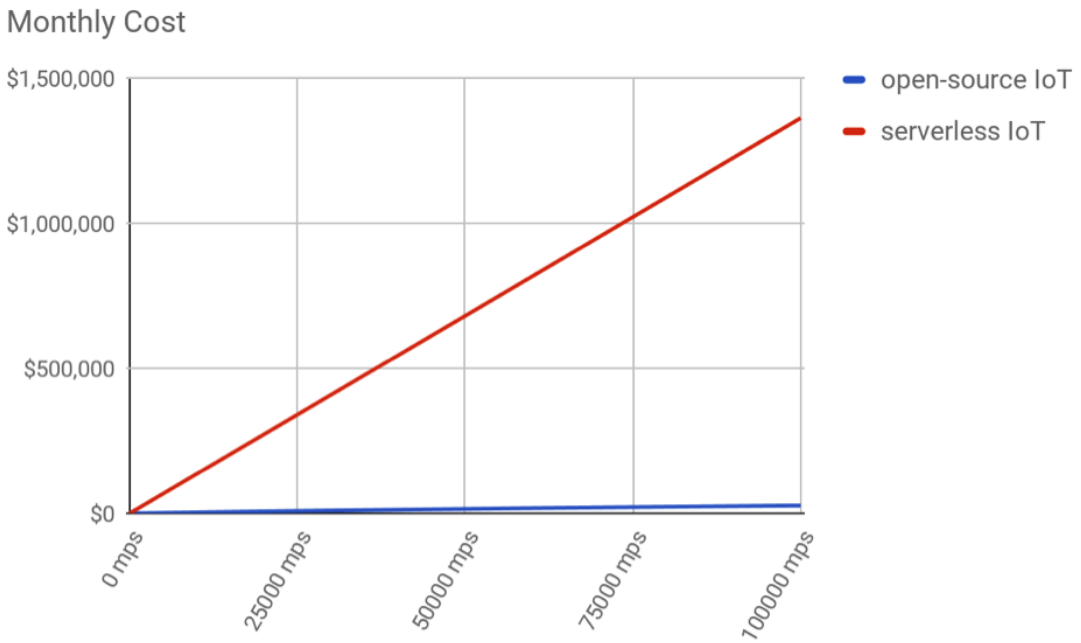
Once an AWS IoT-based system processes beyond 17-18 messages per second, it’s time to start thinking about moving to another IoT platform to reduce operating cost. Moreover, the cost per message drops dramatically after 20 messages per second for open-source IoT platform and continues to decrease with the growing message rate, while remaining constant for serverless IoT.

Let’s look at greater numbers. What happens if we have 10000 devices? We get $136,000 per month for AWS IoT versus $3,300 for open-source! This is indeed impressive, and if switching to another platform did not look so attractive before, now it looks like a must. The more

messaging you have, the less cost efficient is serverless IoT, and the more you can save with another platform.

The difference, which makes almost $1.6 million a year, should be enough to hire an engineering team to operate the product or build custom features, spin up a few extra instances for automatic orchestration, and get into a support contract with the company that develops the platform.

And finally, here is the monthly cost chart for a highly loaded IoT system. You can make your own conclusions.



The case with AWS vividly shows the enormous cost difference between the two approaches. It may not be obvious on small message volumes with other IoT cloud platforms, but it's only a matter of finding that point when serverless approach becomes cost ineffective.

Conclusions

While cloud-based serverless IoT solutions provide vast integration options out of the box and instant time to market, the numbers speak for themselves – be ready for a huge bill.

If the business is not expected to grow fast, or for business concept validation, a cloud-based solution is best. Otherwise, if the IoT platform matches most of the criteria above, then my advice – go with an open-source solution.

Written by Igor Ilunin, Head of IoT at DataArt.

Original article can be found here: https://www.iotforall.com/iot-platform-open-source-vs-serverless/amp/