X-IO technologies
BETTER BY DESIGN

White Paper

IoT Edge Computing:
Architected for Performance and Reliability

No recent technological trend has been as broad in scope and potential impact than the Internet of Things (IoT). The idea connects devices to each other, to computing resources, and to humans in order to customize our environments, detect threats, optimize operations, and make decisions that humans wouldn't be able to make on their own.

The IoT computing resources are often thought of as residing in "the Cloud" – faceless servers in some unknown location that provide, effectively, infinite computing capability. But some problems – in particular, those that need fast calculations for real-time decisions while simultaneously ingesting large quantities of data – need edge computing as well.

The notion of the "edge" varies according to context. For our purposes, the edge implies computing resources at the outer edge of the wide area networks that connect IoT devices to the Cloud. If IoT devices are thought to be "here" and the Cloud is thought to be "there," then edge computing occurs here, rather than there in the Cloud. This eliminates longer transit times and improves performance over Cloud-computing resources that aren't optimized for speed.

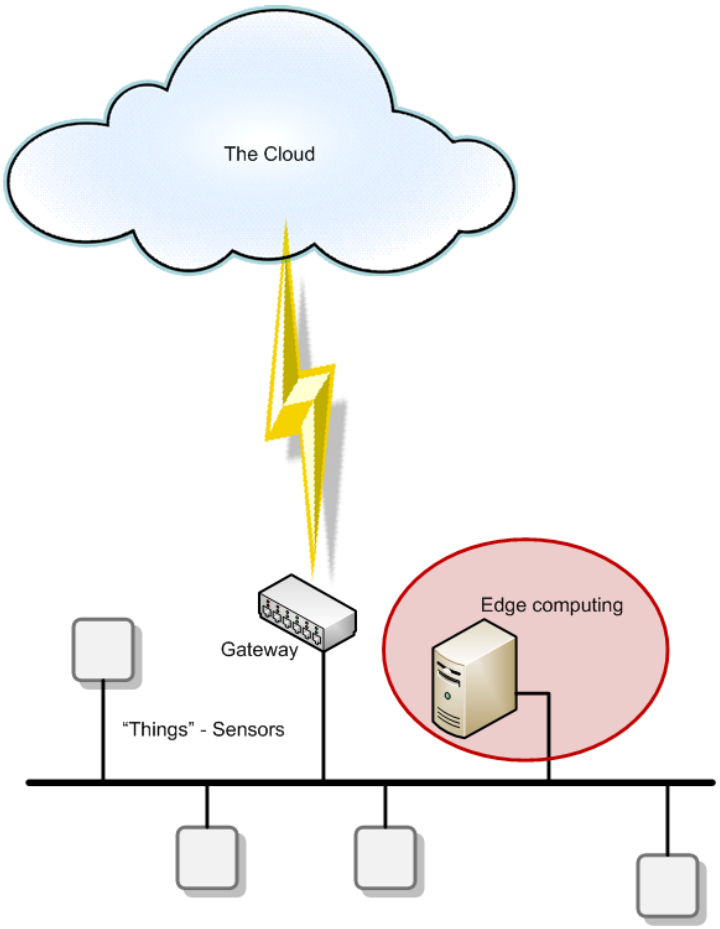

Figure 1. Edge computing provides local resources for real-time decision making

That said, the key notion that defines edge computing is the ability to handle huge volumes of inputs from nearby sensors and digest them with low latency in order to support real-time needs. Any architecture that supports this would qualify as edge computing, regardless of the physical or geographical location of the edge-computing facilities.

Critically, an edge-computing facility is not simply a local Cloud. The Cloud architecture is optimized for scalability and cost, not performance. In order for edge-computing resources to meet the latency and I/O challenges, a very different architecture is required.

The Industrial Internet of Things

While the IoT is often characterized as a single beast, there are multiple embodiments having dramatically different characteristics. The biggest divide is between the consumer IoT (CIoT) – typically characterized by the smart home – and the Industrial IoT (IIoT) – a broad range of applications including factory automation, fleet management, financial markets, smart city infrastructure, and a variety of other spaces.

The biggest difference between the CIoT and the IIoT is that the CIoT focuses on coordinated interactions between devices as well as tracking metrics in the Cloud. The IIoT, by contrast, is about process optimization – whether that means better manufacturing yields, reduced unscheduled maintenance of machines and vehicles, minimizing energy consumption, or tracking of available parking spaces. Because IIoT operations involve large ongoing streams of rapidly changing data that feed heavy calculations, the edge-computing concept is far more applicable to the IIoT.

Examples of applications that can benefit from edge computing are:

- Network protection: this includes threat analysis, intrusion detection and prevention, and traffic monitoring and management. It involves detailed analysis of all packets – potentially going deep into them – at extremely high speeds in order to keep up with the traffic.

- Smart grid: energy companies assess huge numbers of sensors and smart meters. The analysis must be provided in a timeframe that allows utilities to make energy delivery adjustments in response to loading and other events.

- Financial markets: modern trading consumes vast quantities of data both to make investment decisions – where milliseconds matter – and to provide transparency for regulatory purposes.

- Transportation: modern ground, rail, and air vehicles generate an enormous amount of data, some of which will affect the operation of a trip in progress. Naturally, any such decisions must be made as quickly as possible.

- Military operations: warfighters depend on huge volumes of data, whether to provide optimal situational awareness, to guide deployment of assets, or even to optimize the trajectories of munitions in flight.

The IIoT provides efficiencies –replacing machines before they fail to reduce unscheduled downtime; maximizing production yields; optimizing energy distribution. It also helps protect against loss – missing vehicles in a fleet, munitions that miss their targets, or poor financial decisions. And it can improve security if built with the appropriate safeguards.

The common factor in these and other applications is that there is no time to wait for decisions; the inability to manage all of the inputs towards a quick result can, in the worst case, result in physical failure: damaged factory equipment, military misses – even shocks to the financial markets. At best, slow decisions make for suboptimal results.

There are a number of reasons why using the Cloud may not work for real-time decisions. One is the simple issue of delivering huge volumes of data across long distances to access the cloud. The wide area network typically can't handle that much traffic – and even when it can, there is a cost to doing so. And, even if it were free, there's still the speed of light that limits how long a round-trip will take. By doing much of the intensive computing locally, results can be turned more quickly, and less data is sent to the Cloud.

Why Not Use the Cloud Architecture Locally?

If communication contributes significantly to latency, then why not simply build a cloud locally, eliminating the need for data to travel so far? A look at Cloud architecture and philosophy will show why it can't bring latency down enough to support real-time needs.

The Cloud is built on the basis of large numbers of very inexpensive computing nodes (servers) with minimal storage; they are assumed to fail at some point. Because of the low-cost nature of the servers, they won't provide high performance, meaning longer compute times, which means potentially longer latency. Each node will have the bare minimum resources to get by. The focus is on making scaling out – adding (or removing) nodes – as easy as possible.

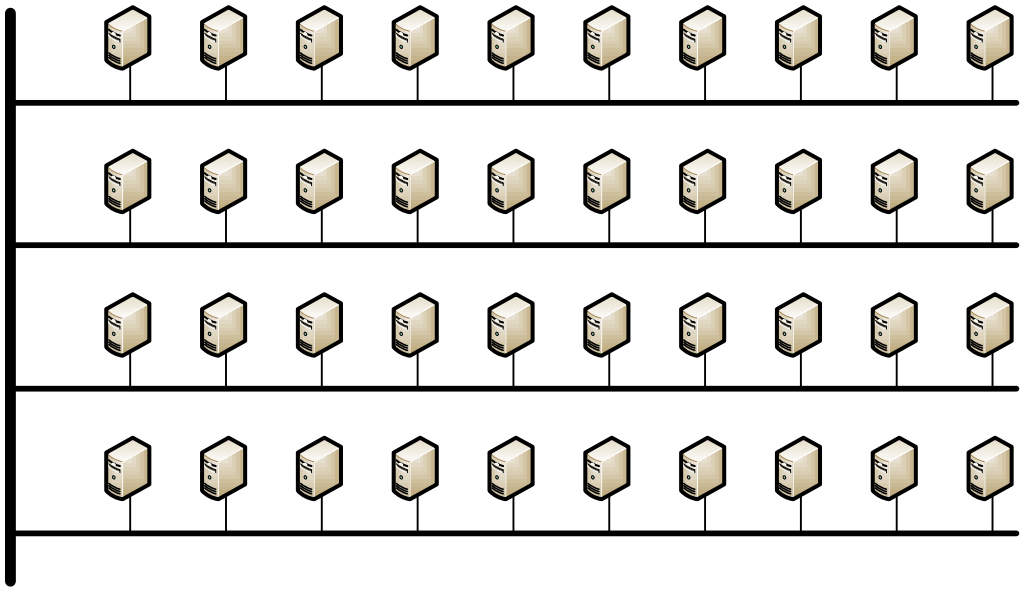

Figure 2. The cloud architecture is based on large numbers of cheap computers, easily scaled, and likely to fail

The Cloud makes up for low-performance servers in numbers: if one server takes too long, then split the problem up and have multiple servers get the job done faster. This can indeed work, but only if the computing problem is well suited to being broken up for parallel execution.

The MapReduce algorithm is one example of a parallel strategy. Instead of one server calculating, say, an average of a huge number of data points (the problems are normally more sophisticated than a simple average), multiple servers can take averages of subsets of the data and then send their partial results to one server that can combine them into a single complete average.

Realistically, however, there are many complex problems that can't be easily split up, so they need to execute on a single platform – which, in a cloud architecture, will execute slowly. Scaling up – that is, moving to more powerful servers – is an option, but the Cloud generally isn't going to host the highest-performance machines.

In addition, because of the susceptibility to failure, Cloud networks must build in resilience. One way of doing that is to distribute data from one node to other nodes so that, if a node goes down, its data can be reconstructed from these copies. The problem is that this creates extra data movement through the network just to support robust operation. The pipes interconnecting the servers therefore provide very high bandwidth – not for supporting in-use data communication, but as overhead to provide robustness.

Finally, because the Cloud is used by multiple tenants, there may be administrative delays associated with availability – especially if node failure necessitates interrupting computation in order to relocate to a new node. These characteristics – limited bandwidth into the Cloud, low performance within the Cloud, and administrative delays – are fine for problems like calculating analytics after the fact, but they are wholly inappropriate when real-time decisions are needed.

Replicating the Cloud architecture locally would eliminate the distance problem, and it might solve the availability issue if there is no contention for resources by competing jobs. But the fact remains that each server is of low performance – and one can't count on parallel execution for speed. So recreating a Cloud locally, while better than a distant Cloud, is still suboptimal.

Edge-Computing Platforms

In contrast to the Cloud architecture, an edge-computing platform will be a single node with enormous computing capabilities, providing:

- High input bandwidth so that the network can deliver the volumes of data constantly being generated;
- High performance to keep ahead of all that incoming data;
- The ability to read stored data for calculations while simultaneously writing new data as it arrives so that no data is lost; and
- Large quantities of local, high-speed storage.

I/O: Incoming data rates can be extremely high. It's likely that multiple 100-Gb/s Ethernet channels will be required.

Processing: Multicore processors are required, with a large number of cores – approaching 100. This is obviously useful for problems that can be parallelized, but, because one can't count on that, these cores won't necessarily be used for splitting up hard problems. Rather, many cores will be doing different tasks, and with different provisioning.

Some could be grouped under an SMP Linux instance for parallel problems. Others may be optimized by running them without an OS at all – like the cores handling packet processing at speeds fast enough to handle the highest volume of the smallest packets.

Working memory: Without sufficient memory, too much time is spent transferring data to and from storage – and this can dramatically slow performance. So the system will need large quantities of RAM – into the multi-terabyte range. Memory should be broadly accessible by multiple processes so as to avoid having to copy data from process to process when they need the same data.

Storage: Enormous quantities of non-volatile storage are needed to capture the torrents of data constantly arriving. Needs can run as high as half a petabyte, with the ability to transfer data on the order of 60 GB/s through NVMe slots.

Internal busses: Communication within the node has to be much faster than the node-to-node bandwidth of the Cloud. The platform may comprise more than one server, in which case the server-to-server bandwidth has to be similar to the incoming bandwidth. For that reason, it becomes difficult to scale the node efficiently beyond two servers.

Reliability: Rather than the assumed-to-fail nature of Cloud nodes, this node has to be extremely reliable – even to the point of redundant power supplies. If it goes down, there's nothing to take its place.

Footprint: An edge-computing node will need to occupy as small a space in a rack as possible, since it's not part of a server farm. A suboptimal solution will take too much space, increasing costs.

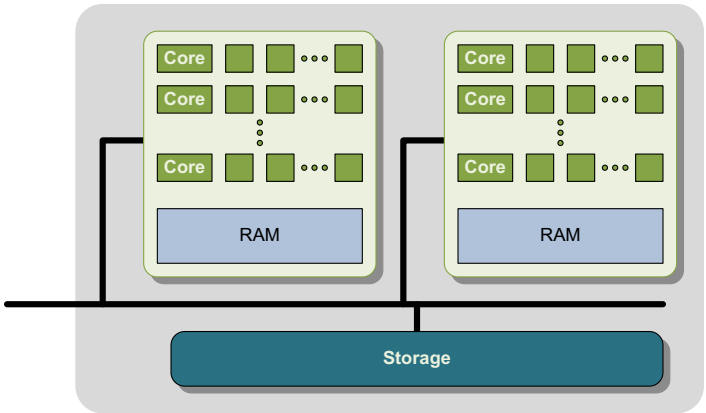

Figure 3. An example of a two-server edge-computing module, with high-bandwidth interconnect, many cores, and ample high-speed RAM and storage

All in all, this is a platform that will have on the order of 30 times the performance of the typical Cloud node.

These are the requirements that have motivated X-IO's FabricXpress™ architecture, which forms the basis of our Axellio™ Edge Computing Systems. They feature:

- 2 dual-CPU servers, with up to 88 cores total
- Up to 2 TB of RAM
- Up to 460TB of dual-ported Flash NVM, shared between the two servers, supporting simultaneous reads and writes
- Over 60 GB/s transfer rates through the NVMe slots
- Expandability and reconfigurability via 6 modular slots supporting any combination of flash storage or additional CPU or GPU offload cards
- Non-Transparent Bridge between the servers for high-bandwidth, low-latency communication
- High-reliability dual power supplies
- 2U form factor
- Size, weight, and power that provide ongoing operational cost savings

Optimizing for the Edge

Complementing the Cloud with an edge-computing facility requires resources tuned to the computation that will occur at the edge. Hard problems that need real-time answers demand a high-performance node – one that doesn't look like a smaller version of the Cloud.

By maximizing network I/O, computing performance, working memory, and storage, an edge-computing node can handle mission-critical real-time requirements, delegating non-real-time duties to the Cloud.

For more information on edge computing or to learn more about our Axellio server modules, contact us at x-io.com.