DZone
A DEVADA MEDIA PROPERTY

Edge Computing and IoT

BROUGHT TO YOU IN PARTNERSHIP WITH

Table of Contents

Welcome Letter

Cate Lawrence, DZone Core Member and Freelance Tech Writer

If you were to pick the most impactful technology of 2020, it would have to be cloud and edge computing. As the world shifted from physical to virtual environments, AWS, Microsoft Azure, IBM Cloud, and Google Cloud underpinned the digital response to COVID-19 and the surge in video conferencing, movie streaming, and gaming. Wearable tech and telemedicine helped to track and monitor those affected by the virus as 62 countries adopted public Exposure Notification Systems using Bluetooth technology.

Companies pivoted to online offerings, and platform vendors profited. Multi-cloud environments have become ubiquitous for global organizations against regional restrictions, service disruption risks, and vendor lock-in.

Google Cloud made Anthos, it's Kubernetes-based, hybrid, and multi-cloud platform on AWS, available to the general public. Anthos allows customers to build and manage applications across environments — in their on-premises data centers, on Google Cloud, and on rival third-party clouds. This is a significant step toward provider partnerships.

It's been a year of combinatorial innovation with "the triple threat" of 5G, AI, and IoT aiding everything from smart cities to autonomous vehicles and health tech. AI converts raw data from the edge into actionable insights. 5G, a key driver for edge computing, has become commercially available globally, paving the way for ultra-low latency, decreased bandwidth, and fast, reliable streaming. Critically, 5G can also monitor cybersecurity in real-time across mission-critical applications.

Cloud-based, as-a-service platforms have made voice and image recognition and chatbots mainstream. Hardware is cheaper and more powerful, and quantum computing makes data analysis faster than ever before. Edge computing capabilities and use cases are evolving, leading to new enterprise opportunities.

We can expect that future applications at the edge will be largely containerized with the help of Kubernetes and GitOps — and the role of developers is critical. Our 2020 Edge Computing and IoT Trend Report provides deep insight into developers' realities at the helm. Developers revealed their experiences with everything from writing software for various IoT communication protocols to the data that operates on the software they build. They share their technical challenges and biggest obstacles to broader adoption of smart objects and devices.

We thank everyone who contributed to the report — survey respondents, authors, editors. And to you, our readers, we hope you can derive actionable insights into the power of edge computing.

Sincerely,

Cate Lawrence

Cate Lawrence, DZone Core Member and Freelance Tech Writer
@Cate_Lawrence on Twitter | @catelawrence on DZone

Cate Lawrence is a Berlin-based tech journalist, writer, and content strategist focused on IoT, mobility, smart cities, emerging technologies, and the relationship between people and tech.

ABOUT

DZone Publications

Meet the DZone Publications team! Publishing Refcards and Trend Reports year-round, this team can often be found editing contributor pieces, working with Sponsors, and coordinating with designers. Part of their everyday includes working across teams, specifically DZone's Client Success and Editorial teams, to deliver high-quality content to the DZone community.

DZone Mission Statement

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community.

We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.

Meet the Team

Lindsay Smith, Publications Manager at DZone

@DZone_LindsayS on DZone | @Smith_Lindsay11 on Twitter

Lindsay is a Publications Manager at DZone. Reviewing contributor drafts, working with sponsors, and interviewing key players for "Leaders in Tech," Lindsay and team oversees the entire Trend Report process end to end, delivering insightful content and findings to DZone's developer audience. In her free time, Lindsay enjoys reading, biking, and walking her dog, Scout.

Melissa Habit, Publications Manager at DZone

@dzone_melissah on DZone | @melissahabit on LinkedIn

As a Publications Manager, Melissa co-leads the publication lifecycle for Trend Reports — from coordinating project logistics like schedules and workflow processes to conducting editorial reviews with DZone contributors and authors. She often supports Sponsors during the pre- and post-publication stages with her fellow Client Success teammates. Outside of work, Melissa passes the days tending to houseplants, reading, woodworking, and adoring her newly adopted cats, Bean and Whitney.

Blake Ethridge, Community Manager at DZone

@FilmFest on Twitter | @blakeethridge on LinkedIn

With twenty-five years of experience as a leader and visionary in building enterprise-level online communities, Blake plays an integral role in DZone Publications, from sourcing authors to surveying the DZone audience and promoting each publication to our extensive developer community, DZone Core. When he's not hosting virtual events or working with members of DZone Core, Blake enjoys attending film festivals, covering new cinema, and walking his miniature schnauzers, Giallo and Neo.

John Esposito, Technical Architect at 6st Technologies

@subwayprophet on GitHub | @johnesposito on DZone

John Esposito works as technical architect at 6st Technologies, teaches undergrads whenever they will listen, and moonlights as research analyst at DZone.com. He wrote his first C in junior high and is finally starting to understand JavaScript NaN%. When he isn't annoyed at code written by his past self, John hangs out with his wife and cats Gilgamesh and Behemoth, who look and act like their names.

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results from DZone's 2020 Edge Computing and IoT Survey

By John Esposito, PhD, Technical Architect at 6st Technologies

From November to December 2020, DZone surveyed software developers, architects, and other IT professionals in order to understand two distinct but interlinked topics — edge computing and the Internet of Things.

We chose to research these two topics together for three main reasons:

1. The rapid influx of data of uncertain value from connected devices has begun to shift the "smart/dumb client balance" back toward "dumb" after powerful web browsers and smart phones had shifted the client balance toward "smart" for the past decade or so.

   The edge computing paradigm seeks to prevent these new dumb clients from overloading central computing nodes. But how to make this happen at a granular level, without introducing unmanageable complexity, seems like a hard problem that would soften under comparative empirical scrutiny.

2. As both smart clients and cloud computing have grown ubiquitous, computing resources have been concentrating at topological endpoints (the browser and the cloud).

   However, this paradigm does not take advantage of the hierarchical abstractions offered by packet-switched networks-of-networks and increasingly necessitated by massive data from dumb devices. The edge computing paradigm takes advantage of these abstractions "in between," but how it does so and what technical problems it solves (and does not solve) are unclear.

3. The homeostatic model of the cyber-physical world most influentially sketched by Norbert Wiener and now embodied by IoT often encodes heavy signal dampening "at the edge" (e.g., dynamic tuning of cochlear bones to focused target frequencies, acute and chronic heat acclimatization, protective reflexes).

   The emerging edge computing paradigm looks like a generalization of this method over modern networked computers. The accuracy and utility of this analogy are still to be determined, as perhaps they have been since the 1950s...

Major research targets:

1. The state of IoT and edge computing — project and device types, data sources and processing structures interacted with, protocol usage, and technical challenges
2. The mind of the edge-computing user and IoT programmer

Methods:
We created a survey and distributed it to a global audience of software professionals. Question formats included multiple choice, free response, and ranking. Survey links were disseminated via email to an opt-in subscriber list, popups on DZone.com, and short articles soliciting survey responses posted in a web portal focusing on integration-related topics. The survey was opened on November 19th and closed on December 4th, recording 638 responses.

In this report, we review some of our primary key research findings. Many secondary findings of interest are not included here. Additional findings will be published piecemeal on DZone.com.

Research Target One: The State of Edge Computing and IoT

The motivations for understanding the state of edge computing and IoT are implied by the reasoning given above for approaching both topics together. In summary, the intersection of a data-locality paradigm (edge computing) and a software-hardware interface paradigm (IoT) is particularly hard to pin down because both are grizzled at the theoretical level but immature at the

implementation level. For this reason, our "state of the field" foci in the present research were fuzzier than in most of our other work. Therefore, in this section, we offer more open-ended, less analytical commentary than usual.

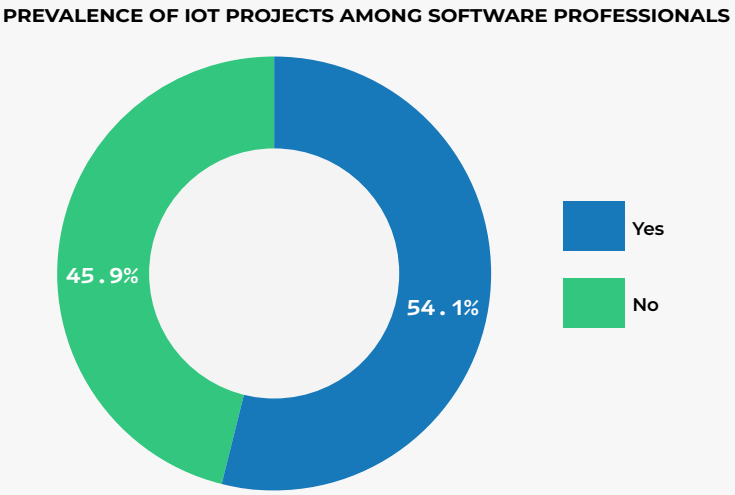
HOW MANY PEOPLE ARE WORKING ON IOT PROJECTS

We wanted to know what proportion of software professionals have been working on IoT projects over time. We already had five years of survey data on this question, so this year we asked:

Have you worked on any projects that you would consider IoT?

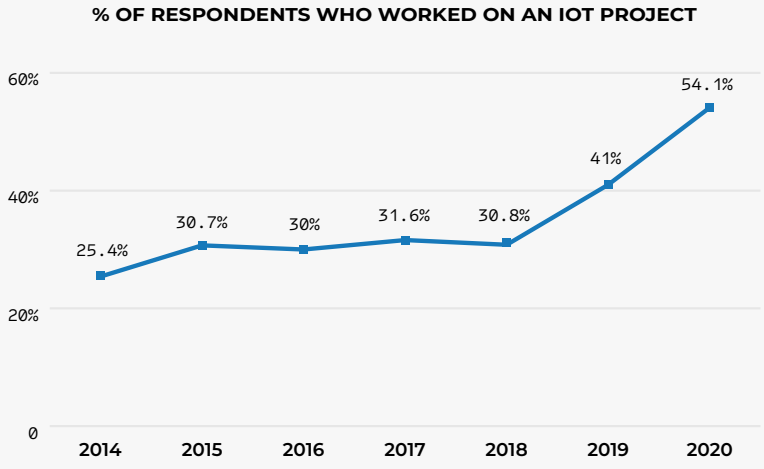
This year, for the first time, a majority (n=336) said yes (total, n=621):

Figure 1



Compare the trend of IoT project prevalence over the past six years:

Figure 2



Observations:

1. Prevalence of IoT project work among software professionals has indeed grown rapidly, especially over the past two years.
2. Last year a significant minority (41%) of developers had worked on IoT projects.

The jump to 54.1% (a 13.1% increase) in 2020 is striking, but it's only slightly accelerated vs. the increase from 2018-2019 (by 10.2%). The similar rate of increase makes both growth numbers — otherwise strikingly high — seem more plausible. In order to

normalize the last two years of data over a changing developer ecosystem and survey population, we analyze some narrower respondent segments under Research Target Two.

NOTE: We did not phrase our question identically in every annual survey. As the industry discourse evolved, we adjusted our question phrasing accordingly. The sums included in this graph represent our best effort to normalize responses. Contact the DZone Publications team if you'd like to discuss details.

DISTRIBUTION OF WORK ACROSS PHYSICAL INPUTS AND OUTPUTS

HYPOTHESIS ONE

Hypothesis: Software is more likely to interact with sensors than with actuators.

Reasoning: Of the two directions of information flow between software and physical worlds, the software → physical direction is the more radical. The analogy-complex that paints software as mind and hardware as brain does not necessarily entail the notion of computer as controller — a loose English translation of the Greek kybernetes, literally "steersman" — the root of the word "cybernetic" in the sense of "pertaining to a control system."

IoT is supposed to add weight to the software → physical side of the scale. But we would like to see how much and how quickly this is happening. We hypothesized that, while at present interaction with sensor data dominates software development, this dominance will weaken over time.

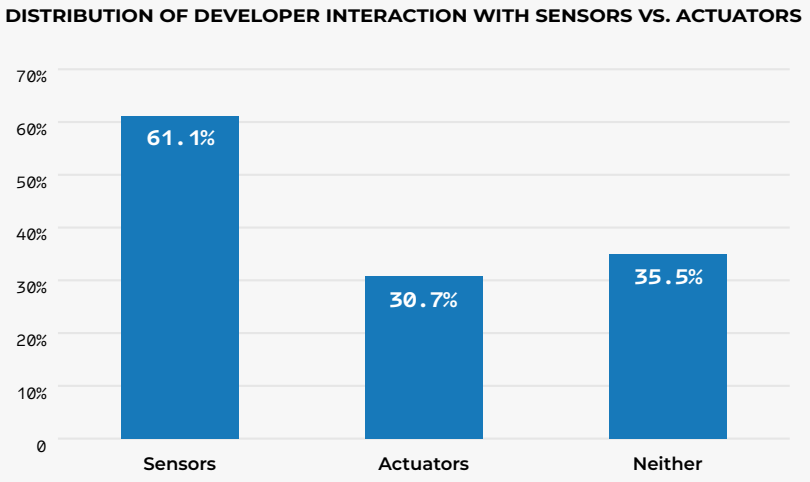
We asked:

Please indicate your experience according to the following statement (check all that apply): I have developed software that controls or otherwise interacts with...

- *Sensors*
- *Actuators*
- *Neither*

Responses were distributed as follows:

Figure 3



Observations:

1. Our hypothesis was verified. However, as noted above, for theoretical reasons this is an easy hypothesis to make. More interesting will be how the sensor:actuator ratio (now roughly 2:1) changes over time. We will ask this question again in future surveys.
2. The sensor:actuator ratio changes slightly between respondents who say they have worked on IoT (45.6% vs. 23.8%; n=277 and 145, respectively: sensor:actuator ratio 1.91:1) and respondents who say they have not (15.5% vs. 6.9%; n=94 and 42, respectively: sensor:actuator ratio 2.24:1).

The small difference is consistent with our historical explanation of the dominance of sensor over actuator interaction in software. Most developers are reasonably likely to have worked with data that in some sense has originated in physical sensors — even if this is as remote as, say, product weight. But many fewer are likely to have worked with data that affects the physical world, except where software falls into the IoT paradigm.

SENSORY MODALITY OF SENSORY INPUTS INTERACTED WITH

The physical world enters the software domain as information encoded digitally, but the immediate object of digital encoding is sensory input of a specific physical modality (e.g., light, sound). In order to understand what informational aspects of the "things" are entering into the "Internet of Things," we wanted to know how sensory inputs of specific physical modalities distribute across developers' work.

We asked:

How has the software that you've worked on interacted with the following kinds of physical data?

	Directly processed sensor signals	*Processed data downstream from sensor*	*None*
Light	☐	☐	☐
Sound	☐	☐	☐
Temperature	☐	☐	☐
Pressure	☐	☐	☐
Moisture	☐	☐	☐
Atmospheric pressure	☐	☐	☐
Electrical current	☐	☐	☐

Summing total checks across both response columns, the ranking of physical modalities interacted with by software respondents have worked on is as follows:

Table 1

Physical Modalities of Source Sensor Most Commonly Worked With		
Modality of physical data	**Rank**	**# of checks**
Temperature	1	436
Light	2	388
Electrical current	3	354
Pressure	4	301
Sound	5	286
Moisture	6	266
Atmospheric pressure	7	235

Observations:

1. The most commonly interacted with physical modality is temperature, by the second largest gap between ranked modalities (48 checks vs. light); the largest gap between modalities (53 checks) is between electrical current and pressure.

   One can imagine that simply displaying temperature in a consumer app pulled from a web API may be counted as "processed data downstream from sensor," and perhaps any data may be considered "downstream" from electrical current. This would make "processed data downstream" responses fairly meaningless.

2. Against this objection, however, is the lack of significant drop from "processed data downstream from sensor" responses to "directly processed sensor signals" for light and electrical current — and indeed for all modalities, as discussed under the hypothesis below.

    199 respondents indicated that they directly processed signals from light sensors vs. 189 who processed data downstream, and 186 indicated that they directly processed signals from electrical current sensors vs. 169 who processed data downstream.

    Since the direct sensory signal processor numbers are slightly higher, these results, if anything, suggest that respondents did not consider the "processed downstream" in the meaninglessly capacious sense we feared above.

3. In future iterations of this question, we will refine the column titles and/or offer additional explanation of what we mean by "processed data downstream from sensor."

    We were specifically aiming to understand the rawness of sensor data processed, with "directly processed sensor signals" as the extreme of rawness, but "processed data downstream" does not distinguish degree of rawness apart from direct DSP.

HYPOTHESIS TWO

Hypothesis: Software is more likely to interact with data downstream from sensors than with sensor signals directly.

Reasoning: Since many applications process data that is in some sense downstream from physical sensors (e.g., one can write a sound driver without directly inferring current variation from microphone membrane pressure), we wanted to distinguish direct processing of sensor signals from the broader pool of downstream, pre-processed data.

We expected that many more respondents would check "processed data downstream from sensor" for each physical modality.

Results (number of checks):

Table 2

Developer Interaction With Data Outputs by Physical Modality of Source Sensor			
Modality	**Directly processed sensor signals**	**Processed data downstream from sensor**	**None**
Light	199 (29.2%)	189 (27.8%)	293
Sound	142 (22%)	144 (22.3%)	359
Temperature	225 (30%)	211 (30%)	359
Pressure	149 (22.5%)	152 (23%)	360
Moisture	134 (20.6%)	132 (20.2%)	386
Atmospheric pressure	116 (18%)	119 (18.5%)	409
Electrical current	186 (27.5%)	169 (24%)	321
TOTAL	11151 (24.7%)	1116 (23.9%)	2395 (51.4%)

Observations:

1. The hypothesis was falsified. We discussed this lack of drop-off above with respect to light and electrical current, but the same holds true across all physical modalities.

2. The vagueness of the question, discussed above with respect to light and electrical current, weakens the overall interpretation of these results.

3. The counter-objection discussed above — that if "processed data downstream from sensor" were too broad, those numbers should be much higher than "directly processed sensor signals" — turns out to be not at all the case.

NOTE: As a result of these observations, we are still unsure how to interpret this data. Contact the DZone Publications team if you have ideas for making sense of these results and would like to access the raw response data.

USAGE OF PROTOCOLS THAT DO NOT REQUIRE THE INTERNET TRANSPORT LAYER

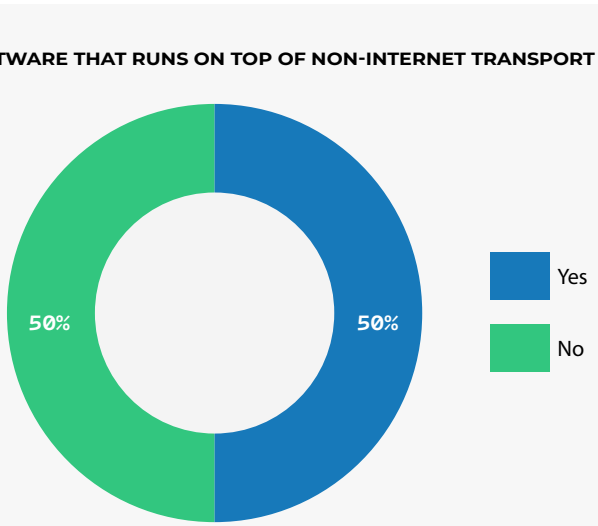
Connected devices "at the edge" of the Internet of Things often do not communicate directly over the Internet. The bridge from the Internet to the leaf-device is therefore built by developers whose code does not run "on top of" Internet transport layers. We assumed most developers write code that runs on devices that do communicate "on top of" TCP or UDP, but we wanted to know how many developers write code that runs on devices that do not.

We asked:

Have you written software that runs on devices that communicate by some means other than a TCP/UDP or some other transport layer that sits on top of the Internet? Consider any communication protocol in your answer, very broadly construed. For example, writing code that talks Bluetooth or Zigbee counts as "Yes" — but so does writing a DSP for sensor data, an audio driver, or code that regulates wattage over a coaxial cable or photons over fiber. These also involve transmitting information without depending on an "underlying" Internet.

Improbably, the results were split exactly evenly (n=614):

Figure 4



Observations:

1. The exactly even split does not appear to be caused by some glitch in the survey software. Responses are not evenly split within segments determined by answers to other questions, and the differences across segments make sense of the current question.

   For instance, far more (63.3% vs. 36.7%) of those who have worked on IoT projects have written code that runs on top of non-TCP/UDP, while far fewer of those who have not worked on IoT projects have done the same (34.2% vs. 65.8%). Again, far more C/C++ developers answered "yes" (72.5%) than JavaScript developers (46.9%). Other sanity-check segments yielded similar results, so we consider the responses collected for this question valid.

2. The overall percentage of developers who have written code that does not use an Internet transport layer is higher than we expected.

   We guessed that the even split might be a function of the overlap of "Internet usage" and "smart device" usage curves. That is, perhaps developers who have not written code professionally before widespread use of the Internet

are significantly more likely to have written code that runs on devices that do not communicate on top of TCP or UDP. This turned out to be true: 66.2% (n=41) respondents with >20 years of professional experience in IT had written software that runs on devices that do not communicate on top of TCP or UDP.

The same group is also significantly more likely (again at 66.1%) to say that they have worked on projects that they consider IoT. Because pre-Internet and IoT eras both involve non-Internet-transport communication, future research might dive deeper into interaction with communication (especially transport) layers over time.

Research might also include correlations of survey data with empirical measurements — perhaps through static code analysis — of protocol usages.

TECHNICAL CHALLENGES TO IOT

As developers, we wanted to understand what makes it difficult to build IoT devices from a technical perspective. We asked:

Please rank the following technical challenges to IoT: {list of answer options; see below}.

Results (sorted by rank from "most challenging" to "least challenging"):

Table 3

Technical Challenges of IoT (Ranked)	
Item	**Score**
Device security	5198
Unpredictable physical environment	4046
Network bandwidth available	3847
Latency	3771
Device unreliability	3507
High volume of unprocessed data	3449
Electrical power (e.g., watt-hours) available on device	3445
Device self-setup/organization	3414
Interoperability	3401
Device discovery	3192
Onboard computing power	3179
Unreliable, undebuggable data	3002
Onboard storage	2711
Extreme data nonlocality	2615
Massive device address space	2378

Observations:

1. Device security was by far the most highly ranked technical challenge.

   The gap between the "device security" score (5198 points) and the second-highest score, "unpredictable physical environment," (4046 points) was almost 4x the size of the second largest gap: "unreliable, undebuggable data" (3002 points) vs. "onboard storage" (2711 points).

2. Respondents who have worked on IoT projects have a significantly different perspective vs. respondents who have not.

   Both groups agreed that device security was the biggest technical challenge to IoT, but those who have worked on IoT

projects ranked “unpredictable physical environment” in second place (with “network bandwidth available” in third), while those who have not worked on IoT environments ranked “network bandwidth available” in second place (with “unpredictable physical environment” in third).

Further, “device self-setup/organization” ranked much higher among those who have worked on IoT projects vs. those who have not (fifth place vs. 12th place, respectively). Again, those who have worked on IoT projects ranked “high volume of unprocessed data” much lower than those who have not (ninth place vs. fifth place, respectively). We might imagine that the technical challenges ranked significantly higher by those who have worked on IoT projects (“unpredictable physical environment”, “device self-setup/organization”) are those who have the thinnest analogies to non-IoT work.

Recent commoditization of computational resources has made immediate computational environments more predictable (consider “five nine” SLAs). Also, recent growth of the REST paradigm for web APIs and sophisticated infrastructure automation (OpenShift, Kubernetes, etc.) have made discovery/registration/organization more self-contained.

Perhaps many developers’ mental habits are formed by these advances, but these advances have not (yet) affected nature or (as strongly) non-HTTP devices.

Research Target Two: The Mind of the Edge-Computing User and IoT Programmer

Motivation:

1. The theoretical pictures of data locality and physical control systems are as old as the von Neumann architecture and the Ashby homeostat. The physical realization of these ideas is perhaps as old as the cell, or perhaps even the general notion of diffusion across concentration gradients. But the modern implementation of these ideas in software flies under the “edge computing” and “IoT” buzzword-flags.

   When developers build working software, however, the marketing veneer peels back quickly. We wanted to know how the fundamental ideas and the buzzwords are differentially expressed in the minds and work of modern software professionals in order to better understand the actual problems still underneath.

2. Design decisions aimed at optimizing locality depend on details of available infrastructure. For example, it is now plausible that on certain devices, a 5G connection might make data available faster than local disk retrieval, meaning the network is no longer the bottleneck. This makes platforms like Amazon Luna (where real-time games are streamed from the cloud) possible.

   The edge computing design addresses the locality problem. We wanted to understand at a high level where infrastructure bottlenecks, especially runtime bottlenecks, are now located.

3. Distribution of computational work optimizes locality but introduces architectural complexity. In particular, state maintenance, including consistency, in distributed systems becomes a non-trivial problem — perhaps the dominant problem.

   In a general sense, IoT distributes computational work (even if predominantly signal processing) extremely broadly, and the paradigm of edge computing tries to abstract over this radical distributed-ness. We wanted to know how developers are solving the distributed state problem especially, but not exclusively, at the connected device scale.

For this research target, we generated some *a priori* hypotheses from theories that reflect our experience in software development. Where our research was designed to test any hypothesis, the hypothesis is presented along with the theoretical reasoning that engendered it. Extra commentary is offered where hypotheses were falsified.

WISHES FOR RUNTIME INFRASTRUCTURE

How do you know when you need better stuff to run your code on? You can answer this with or without an intervening mental processing layer. Without intervening minds, you could run a detailed empirical analysis of stresses, tolerances, and failure rates. This is extremely valuable, but for computational infrastructure, it is presumably best handled by the biggest infrastructure providers — whose skin is maximally in this game, of course, but whose data are obviously not published.

We approached this question by standing on the shoulders of implementing developers' minds. We wanted to know how to improve the car, so we asked the drivers. We also wanted to know how current relative needs for runtime upgrades compare with how relative needs have been in the past.

We asked:

Which of the following do you most want to be available at runtime to the software that you have built most recently (1 = want most, 5 = want least)?

And:

Which of the following do you most want to be available at runtime to the set of all the software that you have ever built (1 = most want, 5 = least want)?

Results:

Table 4

Desired Runtime Infrastructure Improvements			
Software built most recently		**Software ever built**	
RUNTIME NEED	**RANK**	**RUNTIME NEED**	**RANK**
Faster network connection	1	More computing power	1
More computing power	2	Faster network connection	2
More RAM	3	More RAM	3
Larger persistent data store	4	Larger persistent data store	4
More available electrical power	5	More available electrical power	5

Observations:

1. Overall, the variation between current and past software needs is different at the top but the same at the bottom.

   Software built most recently needs a faster network connection more than additional computing power; software ever built needs more computing power than a faster network connection. This is consistent with the standard "probably the network" bottleneck-finding heuristic and, of course, also consistent with Moore's Law.

2. Primary programming language significantly affected ranking, as might be expected.

   C/C++ developers most want more RAM (top ranked) and a larger persistent data store (ranked second). Both results are consistent with writing lower-level code on resource-constrained devices.

   Java developers most want more computing power (top ranked) and a faster network connection (ranked second). Both results are consistent with writing server-side code that runs complex jobs over large datasets and does not worry about memory management.

   JavaScript developers most want a faster network connection (top ranked) and more computing power (ranked second). The former is consistent with the web-focused domain; the latter is consistent with interpreted-ness.

HYPOTHESIS THREE

Hypothesis: People who have worked on IoT projects are more likely to rank available electrical power higher on their runtime infrastructure wants list.

Reasoning: Electrical power is rarely the bottleneck for non-IoT developers. (Note that cost per watt, which someone writing any code that runs at significant scale might worry about, is not relevant to this question. Simply having more power available does not change the cost per watt.)

Results:

Table 5

Desired Runtime Infrastructure Improvements by IoT Project Experience for Software Built Most Recently			
Have worked on an IoT project		**Have not worked on an IoT project**	
RUNTIME NEED	**RANK**	**RUNTIME NEED**	**RANK**
Faster network connection	1	Faster network connection	1
More computing power	2	More RAM	2
More RAM	3	More available electrical power	3
Larger persistent data store	4	More computing power	4
More available electrical power	5	Larger persistent data store	5

For software ever built, rankings were identical between those who have worked on an IoT project and those who have not, with "more computing power" on top and "more available electrical power" on bottom.

Observations:

1. The hypothesis was falsified. In fact, surprisingly, those who have worked on an IoT project ranked "more available electrical power" as their lowest runtime infrastructure desire, while those who have not worked on an IoT project ranked "more available electrical power" second lowest. This is not only contradictory but also contrary to our expectation. Some possible explanations:
    - Developers who have not worked on actual IoT projects (like our primary survey author) magnify electrical power into a bigger problem than it really is because of our crude intuitive picture of IoT.
    - IoT development is now mature enough to keep excess electrical power usage out of requirements, so IoT developers don't feel power as a constraint.
    - As software professionals, our survey respondents worry about power less than hardware professionals; and those who have worked on IoT projects are more likely to have worked on teams with hardware specialists.
2. In future surveys, we will distinguish "on-device" from "most recently built" and "ever built" categories.

    It is possible that many developers who have worked on IoT projects have not worked on IoT projects most recently, and it is likely that most developers who have worked on IoT projects have worked on more non-IoT projects than IoT projects. These latter two conditions might miss or drown out on-device needs, respectively.

ARCHITECTURAL TRADEOFFS WITH RESPECT TO COMPUTATIONAL RESOURCE LOCALITY

As in all engineering, compute resource topology involves tradeoffs. As in all fashion, edge computing is not suitable for all needs. In both engineering and fads, tradeoffs and faddishness depend partly on the problem, but partly on mental habits. If this were not true, tradeoff selections would be algorithmically determined, and technical solutions would not follow trends. We wanted to understand software professionals' general mental models — their modeling biases — with respect to computational resource locality. This is our big "how do you think about edge computing" question.

However, this question overlaps a huge chunk of software architecture. In order to capture as much thought as concisely as possible and to explore underlying model biases rather than concrete problem-solving ability, we constructed a fairly abstract, but very common, scenario-based question with as many external constraints removed as possible. We also made our answer options quite heterogeneous in order to focus on the mental model used to approach the problem rather than the design of any resulting solution, which varies too much by case to be addressed in a single question.

We asked:

Consider the following scenario: Your system needs to process a large number of simple messages from a large number of uncoordinated data sources and make some decisions based on the aggregate of these messages. You have plenty of

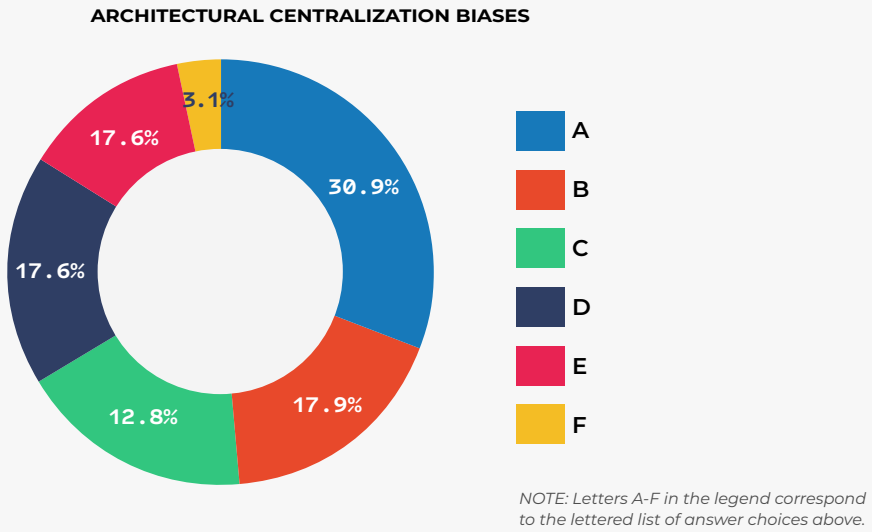
storage space and computing power available no matter how you design your system. After weighing all the pros and cons, no compelling reason forces you into any particular architecture. So it's time to show your biases.

Which of the following best captures your preferred way to handle this scenario? (Remember that we are assuming that the specifics of the problem do not suggest any particular answer.)

A. *Business logic should be defined as centrally as possible, but it should be executed as close to the data sources as possible.*
B. *Business logic should be both defined and executed as close to the data sources as possible; the central node should serve as a proxy, or interface layer, or "dumb sink" only.*
C. *Workload distribution matters much less than architectural simplicity and intelligibility.*
D. *Aggregating functions should be applied as close to the data sources as possible; a central node should work on the outputs of those aggregations.*
E. *The system should contain no central node; a significant portion of the business logic should be implemented in protocols and API contracts.*
F. *Other (write-in)*

Results (n=414):

Figure 5



Observations:

1. The least opinionated option, "business logic should be defined as centrally as possible, but it should be executed as close to the data sources as possible" (30.9%, n=128 responses), was by far the most common response. This is not surprising, as the abstractedness of the scenario suggests an abstracted answer.

2. More interesting is the pragmatism suggested by the least common response, "workload distribution matters much less than architectural simplicity and intelligibility" (12.8%, n=53).

   In our judgement, this is the second least opinionated option, and for reasons mentioned immediately above, we would expect least opinionated options to do better. But it seems that software professionals are more opinionated than this.

3. Selection counts of the three most opinionated options (n=74, 73, and 73) were effectively indistinguishable.

   These three options differ mainly along a "work done by a central node" dimension. It appears that software professionals have definite opinions on the "thickness" of a central node, but no single position on the spectrum dominates.

4. The position of this question in our survey (question #14), after a number of IoT-related questions, might have primed some respondents to focus more on resource constraints despite our in-question stipulation that no system design among the available options will run into storage space and computing limits. In future surveys, we may alter the position of this question to see whether any priming effects might have distorted responses.

HYPOTHESIS FOUR

Hypothesis: Java developers are more likely to prefer "thicker" central nodes than JavaScript developers.

Reasoning: JavaScript developers overall are more habituated to treating clients as "smart" vs. Java developers.

Results (option rankings segmented by Java vs. JavaScript developers):

Table 6

Architectural Centralization Bias by Primary Programming Language			
Java developers		**JavaScript developers**	
PREFERRED APPROACH	**RANK**	**PREFERRED APPROACH**	**RANK**
Business logic should be defined as centrally as possible, but it should be executed as close do the data sources as possible (26.6%, n=46).	1	Business logic should be defined as centrally as possible, but it should be executed as close to the data sources as possible (40%, n=12).	1
Business logic should be both defined and executed as close to the data sources as possible; the central node should serve as a proxy, or interface layer, or "dumb sink" only (21.4%, n=37).	2	Workload distribution matters much less than architectural simplicity and intelligibility (16.7%, n=5).	2 (TIED)
Aggregating functions should be applied as close to the data sources as possible; a central node should work on the outputs of those aggregations (18.5%, n=32).	3	The system should contain no central node; a significant portion of the business logic should be implemented in protocols and API contracts (16.7%, n=5).	2 (TIED)
The system should contain no central node; a significant portion of the business logic should be implemented in protocols and API contracts (16.2%, n=28).	4	Business logic should be both defined and executed as close to the data sources as possible; the central node should serve as a proxy, or interface layer, or "dumb sink" only (13.3%, n=4).	3 (TIED)
Workload distribution matters much less than architectural simplicity and intelligibility (13.3%, n=23).	5	Aggregating functions should be applied as close to the data sources as possible; a central node should work on the outputs of those aggregations (13.3%, n=4).	3 (TIED)
Other (4%, n=7)	6		

Observations:

1. The data are inconclusive. The clearest signal from these responses pertains not to relative system design biases, but rather to relative opinionated-ness on system design.

   JavaScript respondents were significantly more likely to select the least opinionated option (40%, n=12 vs. Java 26.6%, n=46). We take this to mean that JavaScript developers either make fewer system-level compute locality decisions or that fewer system topologies are available, constraining "central node thickness" choice space.

2. However, the small *n* for JavaScript developers who responded to this question makes these answers statistically weak.

RELIANCE ON EXTERNAL DATABASES TO MAINTAIN APPLICATION STATE

Doing more work "at the edge" means inviting distributed systems problems. For non-transacting systems, consistency enforcement need not be maximally local: Central "leader" node may reconcile all workers eventually without any harm. However, since the edge computing paradigm pushes toward keeping work away from central nodes, edge computing tends to take away an easy "leader fallback" option.

In order to understand how well developers' minds are prepared to solve problems in more local, gateway-level "edge" systems, we wanted to know how often developers rely on external adjuncts for state maintenance. We also wanted to know how much developers consider their state-storage choices to be shortcuts.

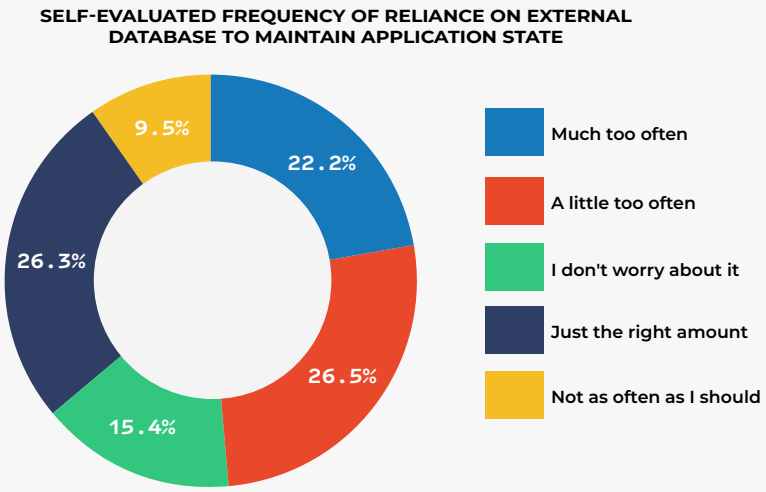
We asked:

How often do you rely on an external database to maintain application state?

- *Much too often*
- *A little too often*
- *I don't worry about it*
- *Just the right amount*
- *Not as often as I should*

Results (n=441):

Figure 6



Observations:

1. Respondents recognize reliance on centralization in the form of reliance on an external database for application state as a temptation toward suboptimal design.

   A minority (41.7%, n=184) did not judge their reliance on an external database excessive. A smaller minority (26.3%, n=116) judged their reliance level correct. The smallest portion (9.5%, n=42) thought they should increase their reliance.

HYPOTHESIS FIVE

Hypothesis: People who think they over-rely on an external database to maintain application state are more likely to prefer "thinner" central nodes than those who think they under-rely.

Reasoning: If you don't enjoy worrying about Byzantine generals, then you may be tempted to take centralizing shortcuts to minimize distributed consistency problems. If you do worry about Byzantine generals, then you aren't so tempted. The larger goal is to understand how closely general architectural biases relate to the specific pattern of centralizing application state.

Results (n=126):

Table 7

Architectural Centralization Bias vs. Reliance on External Database for Application State			
Rely "much too often" on external database to maintain application state		**Rely "not as often as I should" on external database to maintain application state**	
PREFERRED APPROACH	**RANK**	**PREFERRED APPROACH**	**RANK**
Business logic should be defined as centrally as possible, but it should be executed as close to the data sources as possible (28.3%, n=26).	1	Business logic should be defined as centrally as possible, but it should be executed as close to the data sources as possible (35.3%, n=12).	1
The system should contain no central node; a significant portion of the business logic should be implemented in protocols and API contracts (22.8%, n=21).	2	Business logic should be both defined and executed as close to the data sources as possible; the central node should serve as a proxy, or interface layer, or "dumb sink" only (20.6%, n=7).	2 (TIED)
Business logic should be both defined and executed as close to the data sources as possible; the central node should serve as a proxy, or interface layer, or "dumb sink" only (20.7%, n=19).	3	Aggregating functions should be applied as close to the data sources as possible; a central node should work on the outputs of those aggregations (20.6%, n=7).	2 (TIED)
Aggregating functions should be applied as close to the data sources as possible; a central node should work on the outputs of those aggregations (14.1%, n=13).	4	Workload distribution matters much less than architectural simplicity and intelligibility (11.8%, n=3).	3
Workload distribution matters much less than architectural simplicity and intelligibility (13%, n=12).	5	The system should contain no central node; a significant portion of the business logic should be implemented in protocols and API contracts (5.9%, n=2).	4

Observations:

1. The hypothesis was weakly verified. Those who rely on an external database for application state "much too often" ranked the "no central node" option second, while those who rely "not as often as I should" ranked the "no central node" option last.
2. The small *n* for "not as often as I should" respondents (n=34) weakens our inference.

Future Research

This survey covered two large topics, albeit cursorily. For this reason, in addition to individual questions, responses that we have not had space to publish here, we noticed a larger than usual number of cross-question correlations (across edge computing and IoT topics). Additional areas covered in our survey include:

- Transaction vs. stream processing and relation to preference for "edge-ier" topologies.
- Use of event sourcing and strategies for locating aggregation nodes.
- Use of specific mathematical disciplines for software design.
- Use of IoT-suitable ("skinny") protocols.
- Shape of physical sensor data and its impact on architectural decisions.

Follow-up analyses will be published on DZone.com. Follow-up questions will be included in future surveys on software architecture, performance optimization (at both algorithm and system design levels), and embedded development.

ORIGINAL RESEARCH

Leaders in Tech

Muhmammad Rehman, VP and Head of Product for Edge Computing at Verizon, Offers Principal Advice to IoT and Edge Community

Lindsay Smith, Publications Manager at DZone

Muhammad Rehman
VP and Head of Product for CDN, Security, and Edge Computing at Verizon Media Inc.

@muhammadrehman on LinkedIn

Due to COVID-19 and the shift to virtual work around the globe, our reliance on connected devices and the cloud proved unparalleled in terms of demand, growth, and most importantly, data. Our research explored the exceptional benefits of IoT and edge computing, as well as some of the challenges and opportunities for growth.

We sat down with VP and Head of Product for CDN, Security, and Edge Computing at Verizon, Muhammad Rehman, to discuss our research findings, the challenges developers face, and his advice to the broader IoT and edge computing community.

MUHAMMAD'S ADVICE TO DEVELOPERS

- **Practice basic security hygiene.** When it comes to securing your IoT devices, practice basic security hygiene, like updating passwords, disabling unused features, implementing authentication and authorization methods, etc.
- **Protect your IoT projects end to end.** This means protecting against software vulnerabilities as well as securing your physical infrastructure.
- **Adopt a customer-first perspective.** Do so by addressing device use case and latency requirements. This will create overall a more successful and efficient product.
- **Sensors + actuators = the most important partnership.** Sensors and actuators are supposed to be working together and really providing an end-to-end customer experience based on the use case. This will allow IoT devices to collect valuable data that can drive business-critical insights.
- **Embrace the power of 5G.** In the coming months, and even years, 5G is going to play an essential role in the IoT and edge computing space. This technology will allow companies to provide deeper and more immersive consumer products.

What's your advice to developers around securing their devices? And what are the steps that they should be taking?

Technology is evolving rapidly around us, but change is good. As we experience the fourth industrial revolution, things that were impossible a few years ago have become a reality now. One of the essential platforms that is driving this innovation is the global rollout of a 5G network.

I would recommend organizations or developers to take five key steps to secure their devices when starting a new IoT project:

1. Make sure to review and change default passwords and security settings that have been provided by the manufacturer.
2. Only use relevant features and disable any set of features that are not being used.
3. Today's devices are capable of using third-party applications. Make sure to understand the scope of these third-party applications.

4. Perform basic security checks on a regular basis to make sure that the applications that these devices are using will be protected against known security vulnerabilities.
5. Take a holistic view around the authentication and authorization mechanisms for the applications that these devices are using.

What is the best way for developers to handle the unpredictability of their physical environments when working on IoT and edge computing projects?

I would like to categorize the unpredictability of physical environments into two kinds of attacks. First is invasive attacks where an attacker is trying to access and sabotage the physical infrastructure — e.g., the chips actually running these IoT devices are targeted and have been compromised. The second kind of attack is a non-invasive attack where the application or software running on these devices has been compromised.

From an IoT lifecycle perspective, software vulnerabilities are easier to attack and have a specific countermeasure to protect these devices. But when we think about the physical security, it's basically the highest difficulty to crack from an attack perspective. So my guidance to enterprise companies as well as developers would be to focus both on software vulnerabilities and physical infrastructure security to make sure you are protecting an end-to-end solution as you deliver IoT projects.

Another key challenge from our research revolved around latency and bandwidth. How do you address network bandwidth issues for IoT devices?

Most IoT devices use very little bandwidth, but the sheer volume and the expansion of these devices going online means more bandwidth will be needed. As the number of IoT devices grows, we see there's a necessity to make sure that the network accommodates this growth.

We need to make sure that developers and companies are thinking from a customer-first viewpoint, understanding use cases and latency requirements.

In short, network bandwidth is a key concern for IoT projects, but since IoT devices usually use no definite amount of bandwidth with the 5G rollout, I don't see network bandwidth issues becoming a huge problem.

Those who have worked on an IoT project ranked "more available electrical power" as their lowest runtime infrastructure desire, while those who have not worked on an IoT project ranked "more available electrical power" second lowest. We found this surprising, anticipating it to rank higher amongst respondents.

1. Any insight as to why it was ranked so low?
2. What's your general advice to managing a successful runtime infrastructure?

It is quite possible that organizations who have implemented IoT projects have experienced constraints around electric power management for the devices that they have incorporated. And for a lot of organizations who have not embarked on IoT projects, they may not have the complete understanding of how power management actually works within IoT devices and applications.

IoT devices serve a large number of applications. The majority of these devices are battery-powered devices that can be easily installed anywhere, and they are very small in terms of footprint. When we talk about battery life performance, it really depends on the use case. There may be use cases around battery life requirements, which can range from a few hours (e.g., applications like smartwatches) to a number of years (e.g., for things like smart meters or thermal sensors).

As more and more organizations start implementing IoT projects, they will realize that electrical power management for IoT devices is a key focus area and they need to implement optimal power consumption, like identifying active versus deep sleep mode so that less power is consumed. Organizations can also look at different power-saving features, application behavior, and interaction with wireless networks.

Lastly, a lot of manufacturers, when they are building IoT devices, are also trying to look at how to leverage alternative energy sources besides battery-powered devices so that these devices can be operational for a longer period of time.

When looking at the distribution of work across physical inputs for IoT projects, we found the following:

1. Software is more likely to interact with sensors than with actuators
2. Software is no more likely to interact with data downstream from sensors than with sensor signals directly

    A. What do you think about our findings?
    B. What's the key to successfully managing the distribution of work across these physical inputs?

When you think about different devices working as part of IoT projects, sensors and actuators are supposed to be working together and really providing an end-to-end customer experience based on the use case. In addition to that, the controller plays a critical role where a sensor may collect information and route to a control center, which eventually defines the logic that dictates decision-making. As a result of that, the sensor and actuator work together but also enable the controller to sit in the middle and provide actionable insights that the customer may be looking for from their IoT project.

Where do you see IoT and edge computing in the next 6-12 months? What's important for developers to keep in mind and look out for?

In the next few years, 5G technology is going to become an essential platform that will enable companies to utilize the full potential of the fourth industrial revolution by delivering interactive and immersive experiences across many use cases and industry verticals.

IoT and IIoT are the key use cases that are powered by 5G networks. According to a research report published by Cisco this year, the number of devices connected to IP networks will grow from 18 billion in 2018 to 29 billion by 2023. What's interesting is IoT and machine-to-machine connection will grow from 32% to 50% in total device capabilities, which translates into approximately 14 billion IoT-driven devices connected by 2023. In the US, Verizon 5G Edge and Multi-Access Computing (MEC) is enabling companies to automate their factory workflows, perform predictive analytics, and deliver AR/VR experiences to their end users in less than 10 milliseconds.

With reference to edge computing, what I see in the next six to 12 months depends on how companies will prioritize their use cases and choose the type of edge compute solutions that suit their needs. This could be device edge, 5G edge, CDN edge, the cloud, or a combination of these technologies, which can be selected based on latency, capacity, and cost requirements.

Lastly, video consumption has accelerated beyond the media and entertainment industries, making up the majority of internet traffic. And this is where products like Verizon 5G and Verizon CDN Edge can really drive a lot of innovation, especially enabling use cases like ultra-low latency solutions, artificial intelligence, IoT, and IIoT.

Any final thoughts?

Enterprise companies and developers should focus on understanding their customers' pain points, review their business solutions, and come up with a strategy to leverage 5G technology to deliver next generation user experience, which will result in overall customer satisfaction and business growth.

Lastly, as consumers become more and more dependent on IoT-based services — whether it's a smartwatch, smart home, connected cars, or connected home appliances — there is going to be more guidance provided from a cybersecurity legislation perspective. Since manufacturers have been historically focused on time-to-market and innovation, security has been treated as an afterthought. So my recommendation would be for developers and enterprise companies to keep an eye on 5G rollout, as well as the new legislation that's going to be rolling out so that you can have better understanding of the impact of security on your upcoming IoT projects.

CONTRIBUTOR INSIGHTS

Are Edge and Cloud Computing the Winners of 2020?

Cate Lawrence, DZone Core Member and Freelance Tech Writer

This time last year, no one could have predicted the impact of COVID-19 on the world, with millions dead or left with lasting disability, businesses closed overnight, schools shut down, and governments issued compulsory lockdowns. Amongst massive job losses, we've seen all industries make the shift from physical to virtual services in one form or another: apps and platforms, pivots to new service offerings, increased use of IoT-embedded machines and robots, all of which have been underpinned by cloud and edge computing.

We've seen the importance of digital health monitoring, location intelligence, chatbots, deep learning, supply chain tracking, and data science at the forefront of efforts to ameliorate the impacts of COVID-19. All of these developments have created an unprecedented global digital transformation. Let's take a look.

Cloud and Edge Computing Brings the Digital World to Us

As our worlds moved from physical to digital, it's clear that digital platforms were one of the wins of 2020 as we watched more movies, attempted online trivia, and attended virtual meetings. In the first half of 2020, Netflix gained 25.9 million paid net subscribers globally, and Zoom saw its revenue skyrocket as second-quarter profits more than doubled due to the coronavirus crisis. All of this digital communication and entertainment was underpinned by the use of cloud computing vendors such as Amazon Web Services, Microsoft Azure, and Google Cloud.

Edge networks have also been critical across numerous use cases such as enabling Content Delivery Networks (CDNs) to provide internet services at scale. By operating on the edge, businesses can respond in real-time to requests from all over the globe without jeopardizing efficiency, speed, or latency. We've seen this in everything from virtual conferences to gaming and video conferences. The greater latency enabled by edge computing reduces lag and down-time, making it possible to render video on a variety of devices across global locations and facilitate software downloads. These increased capabilities also help enterprises manage the growth in customers and user requests.

Healthcare Becomes Smart and Digital

Healthcare providers have made the shift to telemedicine with the use of wearable devices to monitor biometrics such as sleep and heart rate providing rich real-time data. Schripps Research conducted nationwide research of participants who wear a smartwatch or fitness tracker device creating DETECT (Digital Engagement and Tracking for Early Control and Treatment), a study to monitor resting heart rate sleep, steps, and also allow people to record symptoms like fever or coughing. They found it possible to predict with a high level of accuracy who has COVID-19 based on sleep, activity, and resting heart rate changes.

Numerous healthcare locations have also deployed robots designed to kill the COVID-19 virus and other pathogens by sweeping rooms with pulsating beams of high-intensity UV light.

Cloud Computing Meets AI to Support COVID-19 Research

Combined efforts have greatly increased COVID-19 R&D with supercomputing increasing data analytics. The COVID-19 High-Performance Computing Consortium brings together the Federal government, federal agency partners, Department of Energy labs, industry, and academics to provide access to the world's most powerful high-performance computing resources in support of COVID-19 research.

Image Source: https://en.m.wikipedia.org/wiki/Summit_(supercomputer)

The Consortium brings an unprecedented amount of computing power — 16 systems with 165k nodes, 600 Petaflops, 6.8m CPU cores, 50k GPUs and counting. This includes Summit (OLCF-4 i), a supercomputer developed by IBM, which is the fastest supercomputer in the world, capable of 200 petaFLOPS.

High-performance computing systems allow researchers to run very large numbers of calculations in epidemiology, bioinformatics, and molecular modeling. They make experiments possible that would otherwise take years to complete on traditional computing platforms. The Consortium is currently focused on helping researchers to identify therapies for patients afflicted by the virus, made possible by the greater volume of COVID-19 data available than when first founded.

Also lending its support is the Folding@home Consortium, made up of 11 global laboratories studying the molecular structure of diseases. It's supported by distributed computing power — anyone can donate their computer's processing power to run physical simulations of protein folding. By simulating how proteins take shape, folding@home is able to understand how diseases act and design drugs to fight them. The project is added by the GPU power of Nvidia and Arm software.

Many Jobs Become Obsolete

If there ever was a year that shows us the potential for a future when technology takes our jobs, it was 2020. Companies closed call centers, replacing human customer service agents with chatbots. Digital payment systems replaced cashiers and toll booth collectors. Robots replaced cleaners.

Image source: https://www.elenium.com/products/touchless-self-service/

Airport kiosks were developed using biometrics, sensors, and voice recognition technology, eliminating the need to touch surfaces and operating using verbal commands and head movements. The kiosks also evaluate vitals, including temperature, heart, and heart respiration rate, and can ask questions as part of a health check process.

Surveillance and Monitoring at the Edge

We've long seen the use of thermal imaging, infrared cameras, Bluetooth beacons, wearable tech, and mobile phones to support employees working in extreme environments, including high heat, toxic gas, open flames, and near heavy machinery. The tech is now being used with specialist software to enable safe return to work in many organizations.

Image source: Video screenshot at 02:25 (under heading) from https://www.ibm.com/products/iot-safer-workplace/details

IBM's Maximo insights enables workplaces to utilize advanced analytics and near real-time access to data from cameras to provide monitoring and predictive analytics. Sensors and cameras can respectively monitor temperatures, occupancy, crowd density, and social distancing. Facial recognition tech can be used by edge-based face mask inspected software that can identify when masks are not worn and notify the appropriate staff members.

Tracking movement makes it possible to see if there are hotspots where masks are not worn and identify areas where people get too close to each other, requiring traffic flow changes. Employees receive alerts on their mobiles or wearables when they are too close to other employees assisting with workplace compliance.

Additionally, many cities have deployed temperature checks and great location intelligence at train stations tracking people traffic and adjusting schedules to reduce overcrowding. Apps can also advise passengers which carriages are the least crowded in real-time and alternative modes of transport to increase safety.

Drones have been used in China to spray disinfecting chemicals in public spaces. In China, Malaysia, and Spain, drones equipped with facial recognition technology monitor crowd compliance and social distancing in real time — not only photographing those who fail to comply (China) but issuing verbal commands to those failing to socially distance.

Enterprise IT Spending on Public Cloud Computing Continues to Increase

Sid Nag, research vice president at Gartner, asserts that the pandemic has validated cloud's value proposition:

> *"The ability to use on-demand, scalable cloud models to achieve cost efficiency and business continuity is providing the impetus for organizations to rapidly accelerate their digital business transformation plans. The increased use of public cloud services has reinforced cloud adoption to be the 'new normal,' now more than ever."*

Gartner asserts that worldwide end-user spending on public cloud services is forecast to grow 18.4% in 2021 to total $304.9 billion, up from $257.5 billion in 2020.

Conclusion

With the ubiquity of cloud computing and the demonstrable value of the edge, we can expect new and valuable use cases in response to COVID-19. The introduction of vaccines requires supply chain and temperature tracking (with the potential of added technology such as distributed ledgers to ensure each batch's sovereignty) and virtual passports and databases to monitor the vaccination status of those who want to travel. This world is forever changed, and digital-first is our new reality.

Cate Lawrence, DZone Core Member and Freelance Tech Writer
@Cate_Lawrence on Twitter | @catelawrence on DZone

Cate Lawrence is a Berlin-based tech journalist, writer, and content strategist focused on IoT, mobility, smart cities, emerging technologies, and the relationship between people and tech.

IoT and Edge technologies are rapidly becoming essential to delivering innovative products and solutions

Redapt's IoT and Edge Practice simplifies the architectural, operational, and security challenges of delivering sophisticated compute and data solutions.

Learn More

Enterprise Infrastructure

Kubernetes Open Source

Security

DevOps

Logistics

redapt | DELL EMC

CASE STUDY

Case Study: Mojix

Anthos at the Edge

CHALLENGE

When Mojix, a leading software company, was developing the next generation of its retail edge platform, the company needed a way to manage thousands of in-store applications across the globe. Mojix was working on a security and supply chain software stack that could be deployed at thousands of retail locations for clients. These stacks acted much like micro datacenters, which meant Mojix needed a way to manage the stacks efficiently. Mojix had only recently moved from VMs to Kubernetes, which meant it needed a solution that could be relatively easy to onboard.

SOLUTION

Redapt developed a proof-of-concept solution built upon Google Cloud's Anthos due to its native Kubernetes support and its ability to scale the management of clusters across thousands of micro datacenters. Mojix was able to build a custom tech stack with the confidence of knowing its platform was available to handle the future management and operational needs of its customers. Mojix adopted Anthos as a foundation of its edge solution product to help deliver Kubernetes across customer platforms.

RESULTS

With Redapt's help, Mojix was able to move forward with its edge solution confidently, knowing it would have the ability to manage its micro datacenters at scale. Through the proof-of-concept results, Mojix gained intel into its Anthos deployment and integration capabilities, and the confidence to move forward with its ambitious edge-to-cloud solution.

mojix

COMPANY

Mojix

COMPANY SIZE

200+ employees

INDUSTRY

Retail

PRODUCTS USED

Redapt Anthos

PRIMARY OUTCOME

Mojix's vertical cloud technologies — fully integrated and managed by Redapt — affords retailers winning edge solutions built for continuous digital transformation.

"A lot of our success, and our customer success, is rooted in our cloud-native solutions that are IoT-enabled and powered by vertical cloud technology from the Google Cloud Platform with Anthos. We also rely on Intel's end-to-end hardware innovations, and our great relationship with Redapt as an edge service partner."

— **Gustavo Rivera**,
Mojix Senior VP of Software Engineering

CONTRIBUTOR INSIGHTS

Using Edge Analytics and Cloud Computing

Design Better Industrial Solutions

Kaumil Desai, Delivery Manager at VOLANSYS Technologies

Connected applications and systems are moving to the cloud faster than ever before through the implementation of IoT. Parallelly, the number of end devices and data generated on the cloud is also increasing. Sensors, mobile devices, wearable technology, and many other connected devices in the IoT ecosystem generate a huge amount of decentralized data. Lack of reliable connectivity, delays, and difficulties in processing large data quantities on the cloud, therefore, raised a challenge in analyzing and extracting important insights from this data.

To overcome this challenge, enterprises are leveraging edge analytics with the help of cloud computing. This will bring stability to IoT networks by bringing computational power nearer to the data source and reducing the delays in analytics, thus resulting in instantaneous vision and resolutions. Because of this, edge analytics is able to create algorithms with the data and provide important insights.

How Analytics Works on the Edge

With the advancement in semiconductor technology, MCUs and processors are equipped with more processing power, specialized hardware components, and computation capabilities, helping to provide faster analytics on the edge by deploying advanced machine learning methods such as deep neural networks and/or convolutional neural networks.

A model developed on popular frameworks like TensorFlow, Keras, and Caffe can be deployed after optimization to run on inference devices like Android and micro-controllers. Inference engines designed considering capabilities of MCUs, like TensorFlow-Lite, TensorFlow-micro, CMSIS-NN, etc., can execute the quantized model on the edge for faster analytics.

Edge analytics benefits organizations where data insights are needed at the edge. According to the report, "Global Markets for Automotive Sensor Technologies," the average number of sensors used in automobiles has increased from 50-60 to more than 100. As a direct result of increase in sensors, we are now seeing a huge increase in the amount of data provided by these vehicles. Edge analytics in the automotive industry will help companies collect, analyze, and process data in real-time, allowing them to take the necessary actions. Also, intelligent applications like collision prevention, traffic routing, eyes-off-the-road detection systems, etc. can be designed through artificial intelligence and machine learning on the edge.

This ensures optimized asset usage, low maintenance, and passenger safety. Similarly, IoT-powered healthcare devices can collect patient's data. Edge analytics can analyze the collected data without the need for continuous network connectivity. A clinician treating a patient with a mobile device or tablet will be able to enter patient data into the analytics platform at the edge where it is processed and displayed in near real-time. This helps to treat patients faster and reduce visit frequency. Also, it adds a secure layer of computing power between the cloud and the device, thus safeguarding patient data.

How Analytics Works on the Cloud

Having seen the advantages of edge analytics, it is important to understand that it does not replace the cloud but complements cloud computing as it brings data closer to the data source. A few processes will continue to be executed in the cloud, for example:

- **Training machine learning algorithms:** The development of a machine learning algorithm depends on large volumes of data, from which the learning process draws many entities, relationships, and clusters before training the model. This can be carried out on the cloud along with the training model.

- **Processing power and storage capacity:** Unbounded scalability for storage, processing power, and ease of deploying analytics makes cloud analytics non-replaceable. Historic data is stored on the cloud that can be useful in the future as cloud-based analytics operates on a larger variety of data. For instance, it can add historical data to streaming data or analyze the output from all devices using edge analytics.
- **Taking advantage of edge devices:** Being connected to a single cloud enables us to collect valuable insights on edge analytics. Cloud has the means to manage and turn that data into meaningful predictions and analysis.

How Does Edge Analytics Complement Cloud Computing?

Real-time decision-making in IoT systems is still challenging due to factors like latency, bandwidth, power consumption, cost, form factors, and various other considerations. This can be overcome by adding artificial intelligence to the edge.

Consider the following:

LESS UTILIZATION OF DATA BANDWIDTH AND TRANSFER

Shifting large amounts of data to the cloud for processing can consume high data bandwidth and produce a noticeable lag that may harm time-critical applications. To avoid this delay and eliminate dependence on data bandwidth, data processing on the edge can be executed.

ELIMINATING THE NEED FOR CONTINUOUS CONNECTIVITY TO THE CLOUD

In industries like oil, gas, or mining, where company employees work on remote sites far from populated areas, the connectivity is non-existent. In such scenarios, sensors on edge devices, such as robots, can capture data, analyze it, and monitor operating parameters — whether or not they are inside their normal range of values.

REAL-TIME PERFORMANCE WITH FASTER PROCESSING

Edge computing dramatically reduces the amount of data that has to be sent over the network, thereby reducing network congestion and speeding up operation time. Instead of running processes in the cloud, edge computing runs processes on local places like a computer, IoT device, or edge server. By bringing computation to a network edge, long-distance communication between a client and server is reduced and real-time insights are obtained.

ENHANCED DATA SECURITY

As we move closer to the data source and become location-aware, instead of having a security camera stream the content of its video feed up to the cloud to later be analyzed for certain situations (unknown people, objects, etc.), that analysis can be done within the camera itself. Data privacy and security concerns associated with biometric data make it extremely important to only use the data locally on the device and not send it out over a cloud connection.

Conclusion

Cloud computing and edge computing are different approaches that purely depend on the application being implemented. While they don't resemble one another, they complement each other exceptionally well. There can't be a one-fit-all solution for each business scenario. There are key factors like real-time performance, cost of bandwidth, size of data, application complexity, etc. that decide whether to adopt edge analytics, cloud analytics, or both — which is ultimately the best of both worlds.

Kaumil Desai, Delivery Manager at VOLANSYS Technologies
@kaumildesai on DZone

Kaumil Desai has worked as the delivery manager for VOLANSYS Technologies for the last three years. He has vast experience in product development, machine learning on edge, complex algorithm design, and software development for a variety of businesses, including industrial automation, electrical safety, Telecom, and more.

Speed up your deployment with **DK IoT Solutions** and **Connectivity**

digikey.com/iot

Digi-Key ELECTRONICS

SPONSOR OPINION

Processors Scale for Edge Intelligence

By Josh Mickolio, Supplier Business Development Manager at Digi-Key Electronics

If we were to look at common topics in this community several years ago (even earlier this year, really), many discussions around edge computing would be focused on technology constraints, why it will be important, what new capabilities are needed, and which applications would benefit most. We've seen many instances of great innovation throughout the past 15+ years of IoT (or M2M), though many weren't a success despite everyone's best efforts. Because of these failures the discussion continued to be "what if."

Central to this has been the processing capability on the edge; in this case, we are defining "edge" as the end nodes themselves, not the network edge, which is another discussion entirely.

Many architectures for IoT designs are get data (sensor node, other), immediately send that data over a connection to the cloud for storage and analytics, and if needed, act on that data. This model works very well for certain applications or in our labs, but it isn't scalable to the needs of most deployments. We are told to save precious battery life in our hardware designs, only wake up the processor intermittently, turn on a radio and shut it off as soon as possible, and use the lightest connectivity protocol available. But what if you have an application like motor monitoring or the current rise of the "invisibles," where devices are built into the world around us, our clothing, and our transportation?

We don't just want all the data — we want the reason for the data. What is important to me, my business, my health, and my safety?

Optimization means we need smarter devices, better security, and faster decisions. Let's decide what is important to the device using machine learning; let's have rigid security that doesn't chew through my battery. These needs are what ARM Cortex-M33 processors have been designed to give us. While it was introduced several years ago and there are many options today, this is still a relatively untapped resource to benefit the IoT. The NRF9160 from Nordic Semiconductor implements the M33 for applications along with an LTE-M and NB-IoT modem. Nordic, Dialog Semiconductor, and Silicon Labs have the NRF5340, DA1469x, and EFR32xG21 families, respectively, for truly enhanced Bluetooth/Thread/Zigbee Applications. NXP and STMicroelectronics have their LPC5500 and STM32L5 families of processors to suit any design.

We will always have the "what if" discussion, and the M33 is a step forward to give us the ability to do what couldn't be done before.

ADDITIONAL RESOURCES

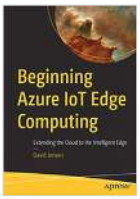
Diving Deeper Into Edge Copmuting and IoT

BOOKS

Fog and Edge Computing: Principles and Paradigms

By Rajkumar Buyya and Satish Narayana Srirama

Want to delve into all things fog and edge computing? Look no further — this book is a comprehensive guide to related applications, architectures, and technologies. Subjects covered include emerging tech, potential research directions, challenges and opportunities in the field, as well as methods applicable to both practitioners and advanced-level learners.



Beginning Azure IoT Edge Computing: Extending the Cloud to the Intelligent Edge

By David Jensen

This book dives into computing on the edge with practical examples of when and how to leverage the power of "intelligent edge computing." Guided through hands-on exercises and by user-friendly discussion, you will learn the process for creating and deploying an Azure IoT Edge solution.

REFCARDS

Edge Computing

Edge computing aims to solve some of the challenges of cloud computing, especially in situations where latency and bandwidth issues would otherwise put operations at risk. This Refcard provides an overview of the concept of edge computing, explores several use cases, and details how to develop an organization-wide strategy for adoption.

Getting Started With Cyber Ranges: Simulated Environments for Cybersecurity Training

A growing number of bootcamps and programs are providing modern approaches to practical skills training, such as hands-on learning via simulated environments. This Refcard provides a brief overview and history of cyber ranges and quick examples of techniques to use on a live-action website.

Distributed SQL Essentials

Distributed SQL databases combine the resilience and scalability of a NoSQL database with the full functionality of a relational database. This Refcard explores the essentials to building a distributed SQL architecture, including key concepts, techniques, and operational metrics.

TREND REPORTS

Data Warehousing: The Emerging Role of Cloud and Hybrid in Modern Analytics

As the demand for informed business decisions and analytics continues to skyrocket, so does the use of data warehouses. This Trend Report explores DW adoption, key challenges, and common data tools and strategies. Readers will find original research, an exclusive interview with "the father of data warehousing," and DZone contributor insights.

The Database Evolution: SQL or NoSQL in the Age of Big Data

Organizations are working to strengthen their big data capabilities to compete in the modern economy. This Trend Report explores the role of popular database types in these big data initiatives. Readers will find DZone research analyses, interviews with industry experts, and contributor content covering new perspectives, strategies and techniques, and more.

PODCASTS

IoT For All

The most creative, intelligent minds in the IoT industry convene on this podcast to share their knowledge and experiences with the ultimate goal to educate others in the field about the inner workings and benefits of IoT.

Over the Edge

Check out this podcast to learn more about edge computing from those who are creating the future of the Internet. Topics discussed include tech innovation, trends, practical applications, and the "occasional far-flung theory."

ZONES

IoT The Internet of Things (IoT) Zone features all aspects of this multifaceted technology movement. Here you'll find information related to topics including IoT, including Machine-to-Machine (M2M), real-time data, fog computing, haptics, and open distributed computing.

Big Data The Big Data Zone is a prime resource and community for big data professionals of all types. We're on top of all the best tips and news for Hadoop, R, and data visualization technologies. We also give you advice from data science experts on how to understand and present that data.

INTRODUCING THE

IoT Zone

The Internet of Things Zone features all aspects of this multifaceted technology movement, extending beyond home automation to include wearables, business-oriented technology, and more.

Keep a pulse on the industry with topics such as:

- Machine-to-Machine (M2M)
- Real-time data
- Fog computing
- Open distributed computing

VISIT THE ZONE

TUTORIALS

CASE STUDIES

BEST PRACTICES

CODE SNIPPETS