INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS

INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS

2023 UPDATE

APPLICATIONS BENCHMARKING

ACKNOWLEDGMENTS

Thanks goes to the members of the team, who are:

Name	Representing	Region
Tom Conte [co-chair]	Georgia Institute of Technology, USA	US
Natesh Ganesh	University of Colorado / NIST, USA	US
Vladimir Getov [co-chair]	University of Westminster, UK	EU
Yoshihiro Hayashi	Keiko University, JAPAN	Japan
Masatoshi Ishii	IBM Tokyo Research Laboratory, JAPAN	Japan
Takeshi Iwashita	Hokkaido University, JAPAN	Japan
Siva Rajamanickam	Sandia National Laboratories, USA	US
Vijay Janapa Reddi	Harvard University, USA [MLperf representative]	US
Masaaki Kondo	Keio University, JAPAN	Japan
Tushar Krishna	Georgia Institute of Technology, USA	US
Peter M. Kogge	University of Notre Dame, USA	US
Scott Koziol	Baylor University, USA	US
Dam Sunwoo	Arm, Inc.	US
Josep Torrellas	University of Illinois at Urbana-Champaign, USA	US
Peter Torelli	Chair, EEMBC	US
Rio Yokoda	Tokyo Tech, JAPAN [SDRJ representative]	Japan

Table of Contents

List of Figures

List of Tables

APPLICATION BENCHMARKING

1. PREAMBLE TO THE 2022 EDITION

The Applications Benchmarking chapter of the IRDS was started as a "top end" to the roadmap. Its mission was to identify key application drivers and to track and roadmap their performance. This activity resulted in several editions of the chapter since the inception of IRDS in 2016. However, this present edition is the last edition of the chapter in this form. There are several reasons for this change.

First, the original goal to track application performance required the team to find existing application benchmarks for each application area. This was not always successful. For example, the Discrete Event Simulation workloads are not particularly taxing to today's architectures. We documented a case where the entire benchmark being tracked reached asymptotic performance after on-chip caches were expanded in an iteration of Intel's i3/i5/i7 microprocessor generations. Another example is Graphics VR/AR. While this application area is important to the industry, the area is not the primary drivers of systems architecture performance. Rather, AI is the dominant application area that drives the GPU platforms shared with Graphics VR/AR.

As another example, Optimization turned out to be an area that was ill-defined. The AB IFT decided to track the performance of near-optimal solvers. But these are not particularly taxing to systems architectures. On the other hand, sat solvers, i.e., true optimization applications, are performance bound not by systems architectures but by the non-polynomial timed algorithms on which they are based.

Second, application areas are forever shifting. We did a slight shift in the last edition of this roadmap to change "pattern recognition" into AI, since the underlying technology, machine learning, had expanded its original borders. This edition takes a new approach that follows the overall IRDS whitepaper approach to new, emerging areas. That is, when there is a new area that the IRDS may track as a potential chapter, it is first introduced as a whitepaper. If the community coalesces around the area, the whitepaper becomes a chapter in the next edition of the roadmap. Similarly, is edition of the Applications Benchmarking chapter provides several emerging areas in a separate section from the prior-tracked application areas. If communities coalesce around these areas, they will become future application areas to cover in this chapter.

What follows is the introduction from the 2020 edition of the AB chapter with some edits to account for the changes as outlined in this preamble.

2. INTRODUCTION

The mission of the Applications Benchmarking[1] (AB) International Focus Team (IFT) is to identify key application drivers, and to track and roadmap the performance of these applications for the next 15 years. Given a list of market drivers from the Systems and Architectures International Focus Team (SA IFT), AB generates a cross matrix map showing which application(s) are important or critical (gating) for each market.

Historically, applications drive much of the nanoelectronics industry. For example, ~25 years ago the PC industry put pressure on semiconductor manufacturers to advance to the next node in the roadmap. Today, as applications shift to the mobile market, it is again manufacturers that are applying pressure for new technology. The market for Internet of Things edge devices (IoT-e) has its own set of pressures and needs, including low cost and low energy consumption. Most of this is discussed in the Systems and Architectures (SA) chapter elsewhere in this roadmap. It is the function of this chapter to step back from the current markets and their needs, and to consider the current and near-future application needs in each of these markets. For this reason, AB was created as part of the International Roadmap for Devices and Systems (IRDS).

The application areas that the AB IFT tracks currently, is proposing to track in future editions, and areas that are no longer tracked, are summarized below in Table AB-1.

[1] *Note that in the computer industry, as opposed to the larger semiconductor industry, "benchmarking" refers to using test programs that serve as proxies for user applications in order to estimate the performance of a computer system on a given application domain.*

Table AB-1 Application Areas

Application Areas Currently Tracked	
Application Area	*Description*
Graph Analytics [formerly "Big Data Analytics"]	Applications of graph algorithms to large static or dynamic graphs
Artificial Intelligence	Modern artificial intelligence applications with emphasis on machine learning approaches. Graphical dynamic moving image (movie) recognition of a class of targets (e.g., face, car). This can include neuromorphic/deep learning approaches such as DNNs.
Physical System Simulation	Simulation of physical real-world phenomena. Typically, finite-element based. Examples are fluid flow, weather prediction, thermo-evolution. Computation is floating-point-based, includes mixed precision.
Cryptography [formerly "Cryptographic Codecs"]	Cryptographic encoding and decoding, including specialized hardware acceleration
Emerging Applications Proposed for Future Application Areas	
Emerging Application Area	Description
Video codec	Encoding, transcoding of video, AOM AV-1
Machine Learning for Science	Use of machine learning techniques for scientific exploration (e.g, graph NNs)
Internet of Things Applications	Applications that run at the edge of the Internet of Things.
Prior Application Areas No Longer Tracked in This Edition	
Discontinued Application Area	Description
Discrete Event Simulation	Large discrete event simulation of a discretized-time system. (e.g., large computer system simulation) Generally used to model engineered systems. Computation is integer-based.
Graphics/VR/AR	Large scale, real-time photorealistic rendering driven by physical world models. Examples include interactive gaming, Augmented Reality, Virtual Reality.
Optimization	Integer NP-hard optimization problems, often solved with near-optimal approximation techniques.

DNN—deep neural network, NN—neural network, AOM AV-1—Alliance for Open Media standard AV-1

In order to track these areas, the AB IFT relies upon existing standard benchmarks where available. These benchmarks should fulfill two criteria:

1. **Benchmark Availability**: There are several benchmark sets available that cover each application area. However, many of these benchmarks either cover only a portion of an application area or cover more than one application area.

2. **Benchmark Results Availability**: In order for benchmarks to be useful for projecting a trend in performance vs. time, there must be a sufficiently-long history of benchmark scores. At a minimum, AB IFT believes at least 4 years prior to the current day of scores should be available.

Some of the application areas studied also have the following requirements:

3. **Metrics vs. Precision and Accuracy**: For some application areas (e.g., feature recognition, optimization), the precision of the result is a parameter for the benchmark performance (i.e., recognizing 81% of faces vs. 85% of faces). Related to this is the accuracy of the result is often also parameterized (i.e., finding a near optimal result that is within 5% of the optimal result).
4. **Power/Energy Scoring**: With a few exceptions, the majority of available benchmarks track metrics that are, unfortunately, disconnected from power dissipation and total energy consumption.

We address these challenges in each of the sections below. For the Cryptographic Codec area, criteria (2) just emerging and being satisfied (sufficient benchmark results are just becoming available).

2.1. Current State of Technology

Application areas of interest for the IRDS were identified through dialogue between the members of both the SA IFT and AB IFT. Graph analytics is a focus of search giants. It is also vital to the business model of social network companies. Artificial Intelligence is increasingly the most impactful application domain for modern systems. It is critical for human-computer interaction and for analysis of non-uniform, irregular data (such as used by the military and by search companies). Physical system simulation is critical to product design in the automobile and aerospace industries (especially for defense applications). It is also of primary importance to the U.S. Department of Energy National Nuclear Security Administration. Optimization is used across engineering disciplines, for example in electronic design automation (EDA). It is also used in consumer products such as global positioning system (GPS) navigation systems. Cryptographic codecs are of high importance in many industries, especially in IoT devices.

2.2. Drivers and Technology Targets

Overall Roadmap Technology Characteristics drivers for Systems (applicable systems or applications attributes) and Support Technologies (applicable device attributes) is summarized below in Section 4 and Table AB-2.

2.3. Vision of Future Technology

All of the application areas of Table AB-1 are expected to remain important over the coming 15 years.

The AB IFT has identified three emerging application areas for potential inclusion in future editions of the roadmap. Video codecs are undergoing a period of rapid development resulting in renewed stresses placed on systems architectures. Machine Learning for Science has emerged as a separate application area with some architectural needs that overlap AI, but with others that are unique and distinct. The application space for Internet of Things devices have expanded to the point where they place stresses on today's embedded computer systems technologies.

As outlined below in Section 4, most of the application areas can benefit from enhanced memory bandwidth. In some cases, benefits will also come from reduced memory access latency. The difference between these is whether the application critically requires data to compute future data (both bandwidth and access latency critical) or not (memory bandwidth critical). A second phenomenon is the use of application-specific accelerators to improve the performance of the application areas, while minimizing overall thermal power dissipation and reducing total energy consumption.

3. Scope of Report

The scope of this report is to characterize the Application Areas, predict their future performance, and to identify technology needs and architectural features. The period of projection is into the next 10 to 15 years. Since this projection is based on the overall evolution of published benchmarks with a long history, this projection takes into account both hardware and software evolution.

4. Summary and Key Points

A summary of the application area analysis is presented in Table AB-2. (Note that Emerging Application areas are not shown.) An entry in this table indicates that the application area is sensitive to the improvement paths shown in the columns. These improvement paths are:

Algorithmic improvement: In this situation, the AB team viewed the algorithm development as an active area that will result in a significant fraction of the performance growth over time.

Memory bandwidth and Memory latency: These two improvement areas are inter-related. There are trends today for improving memory bandwidth onto and off the processors. In addition, memory latency becomes important when traversal is a function of data (e.g., pointer chasing workloads). In general, when the processing being performed is very parallel, memory bandwidth dominates. When it is less parallel (e.g., discrete event simulation or sparse matrix calculations, with the latter included in physical system simulation), then latency is the more important memory attribute.

Network bandwidth: this improvement area relates to the interconnection between processing nodes ("cores"), and the interconnection between processing and memory. This network includes on-chip and off-chip networks.

Fixed-function acceleration: some workloads have highly-repetitive kernels that can be reduced to a specialized, fixed-function data path. For some application areas, this improvement area is expected to give significant benefits in the near future.

Table AB-2 Critical Technology Needs for Application Areas

	Improvement Paths				
Improvement Paths Application Area	Algorithmic improvement	Memory bandwidth	Memory latency	Network bandwidth	Fixed-function acceleration
Graph Analytics		X	X	X	X
Artificial Intelligence	X	X			X
Physical System Simulation	X	X	X	X	
Cryptography		X			X

5. Application Analyses

5.1. Graph Analytics (formerly Big Data Analytics)

This workload class involves a variety of algorithms that operate on a large graph to identify a certain property. The metric that is relevant is computational capability (speed). Ideally, speed would be measured as time to reach the solution of the problem. However, the Graph 500 initiative[1] accepts as a speed metric the traversed edges per second (TEPS) for the problems it considers. A second metric is the energy efficiency in doing these computations, measured in TEPS/watt. This second metric is the focus of the Green Graph 500 initiative[2].

This domain is very popular, and there are a large variety of frameworks. Some frameworks use a shared-memory model and hence run on relatively modest sized servers[3], while others use a distributed-memory model and target very large clusters[4]. Among the frameworks using the shared-memory model, there is no available cross-comparison. But, it is becoming clear that current benchmarks for these frameworks are not benchmarking the whole range of operations that appear in graph workloads. Specifically, current benchmarks do not appropriately cover operations on property-rich graph vertices/edges, and on time-changing graphs. We expect that current benchmarks will change with time. On the other hand, among the efforts that target large, distributed-memory machines, there is an agreed set of benchmarks, namely those in the Graph 500 website. Moreover, the Graph 500 competition has been going on since 2010, and hence there is historical data. For these reasons, we use the Graph 500 benchmark data to track the trends in big data analytics workload performance.

The Green Graph 500 initiative gives a different perspective, as it considers the power consumed in the computation. However, the machines that form the top of this list are small machines. We think they are less representative of this type of workloads. For this reason, we do not consider the Green Graph 500 list.

Up until recently, the Graph 500 benchmark contained two kernels. However, in June 2017, V2.0 added a third one. Since we do not have historical data on the third kernel, we do not consider it. The first kernel constructs a weighted, undirected graph from the given input tuple list, and the second kernel operates on the graph. The first kernel constructs the graph in a format usable by the subsequent kernel. No subsequent modifications are permitted to benefit specific kernels. The second kernel performs a breadth-first search of the graph. Both kernels are timed.

Kernel 1: Graph Construction. The first kernel transforms an edge list to any data structures (held in internal or external memory) that are used for the next kernel. Each edge in the list is a tuple that contains an edge's endpoint vertex identifiers and its weight. Various internal memory representations are allowed, including, but not limited to, sparse matrices and multi-level linked lists.

Kernel 2: Breadth-first Search (BFS). A BFS of a graph starts with a single source vertex. Then, in phases, it finds and labels its neighbors, then the neighbors of its neighbors, and so on. This is a fundamental method on which many graph algorithms are based. The benchmark does not constrain the choice of BFS algorithm itself, as long as it produces a correct BFS tree as output. This benchmark's memory access pattern is data-dependent, with likely a small average prefetch depth. This benchmark measures the ability of the architecture to deliver high throughput while executing many concurrent threads, each with low memory-level parallelism and with a high density of memory accesses. It also measures resilience to hot-spotting when many of the memory references are to the same location. In addition, it measures efficiency when every thread's execution path depends on the asynchronous side-effects of others. Finally, it measures the ability to dynamically load-balance unpredictably-sized work units. The benchmark performs 64 BFS searches from different source vertices executed in sequence. Kernel problem classes are shown in Table AB-3.

Table AB-3 Kernel Problem Classes (the benchmark gives different problem classes, based on the approximate size of the graph)

Problem Class	*Size*
Toy (level 10)	17 GB or around 10^{10} bytes
Mini (level 11)	140 GB or around10^{11} bytes
Small (level 12)	1 TB or around10^{12} bytes
Medium (level 13)	17 TB or around 10^{13} bytes
Large (level 14)	140 TB or around 10^{14} bytes
Huge (level 15)	1.1 PB or around10^{15} bytes

5.1.1. Performance Trends and Prediction

The performance is given in TEPS for Kernel 2. Let *time* be the measured execution time for kernel 2. Let *m* be the number of edges traversed by the search (with some special accounting for some classes of edges called non-self-loop edges). Then, the TEPS (number of edge traversals per second) is given by *m/time*. The output results include additional information, such as the scale of the graph, the Kernel 1 time, various quartiles for the Kernel 2 times (min_time, firstquartile_time, median_time, thirdquartile_time, max_time), and standard deviation values.

Figure AB-1 plots the number of giga-TEPS (GTEPS) of the top six machines from November 2010 (which is when the Graph 500 competition started) to June 2021 (which is the latest measurement). The data is refreshed every 6 months. Nearly all of these runs since 2013 are with the "Large" problem size.

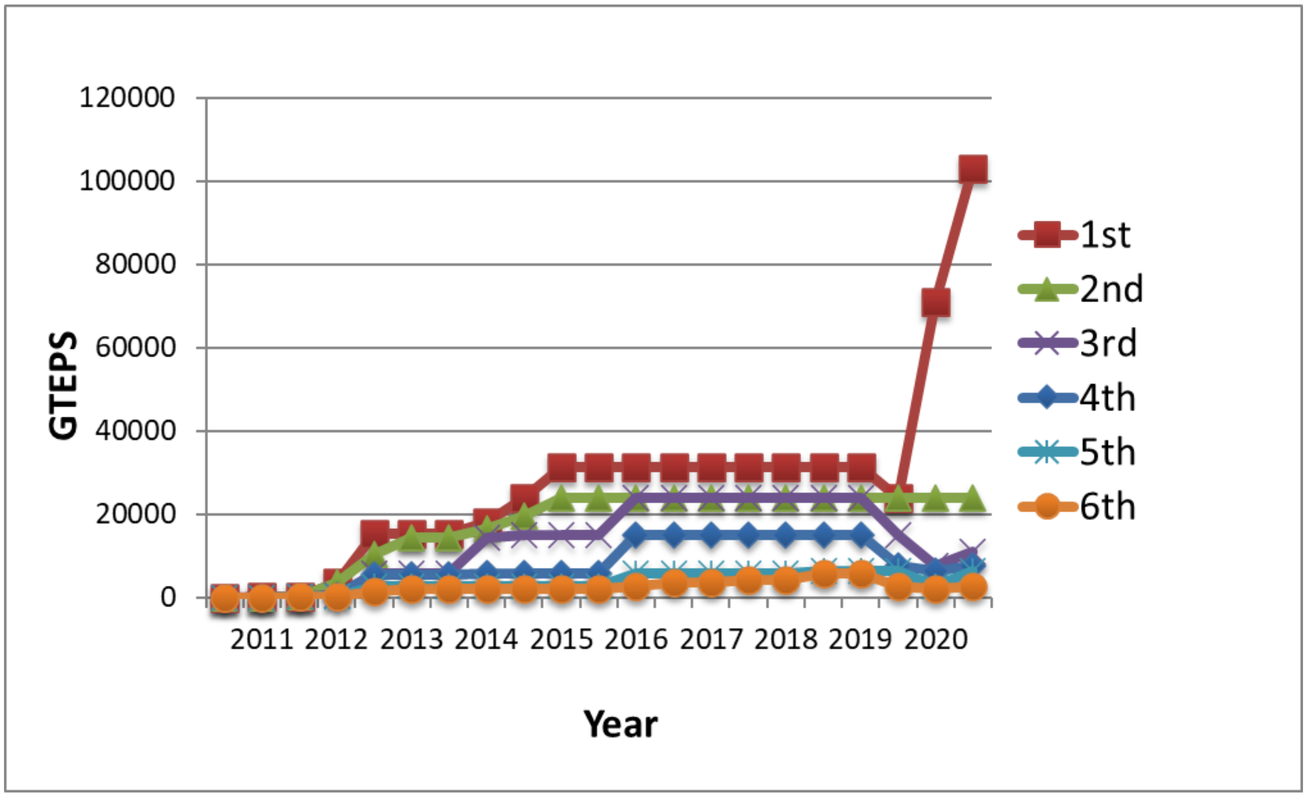


Figure AB-1 Number of GTEPS of the Top Six Machines Since the Inception of the Graph 500 Competition [data via graph500.org]

We see that the GETPS rate has been steadily increasing, although it seems to have been moving in plateaus, until the Fugaku computer appeared in June 2010. As of June 2021, the top three were the Supercomputer Fugaku (Japan), the Sunway TaihuLight (China), and the Wisteria/BDEC-01 (Odyssey) (Japan). Note that the third machine can be regarded as a smaller version of the first one, as it uses the same interconnect and similar manycores. The top three machines are invariably custom-designed machines, with custom networks and, at this point, with 0.25 to 5 Petabytes of memory. The Supercomputer Fugaku has proved to be a game changer in this ranking, at this point providing about a 4x higher GETPS rating that the second machine.

Figure AB-2 plots the number of cores used in the top six Graph 500 machines in the same period of time. If we look at the top three machines, we see that the top two use around 7 to 10 million cores, and the other uses around 350K cores. All three machines use simpler, custom cores---FUJITSU processors based on ARM for the Japanese machines and SW26010 RISC manycore for the Chinese machine), rather than more traditional HPC-class cores.

Overall, we expect that Fugaku will maintain its lead for a while.

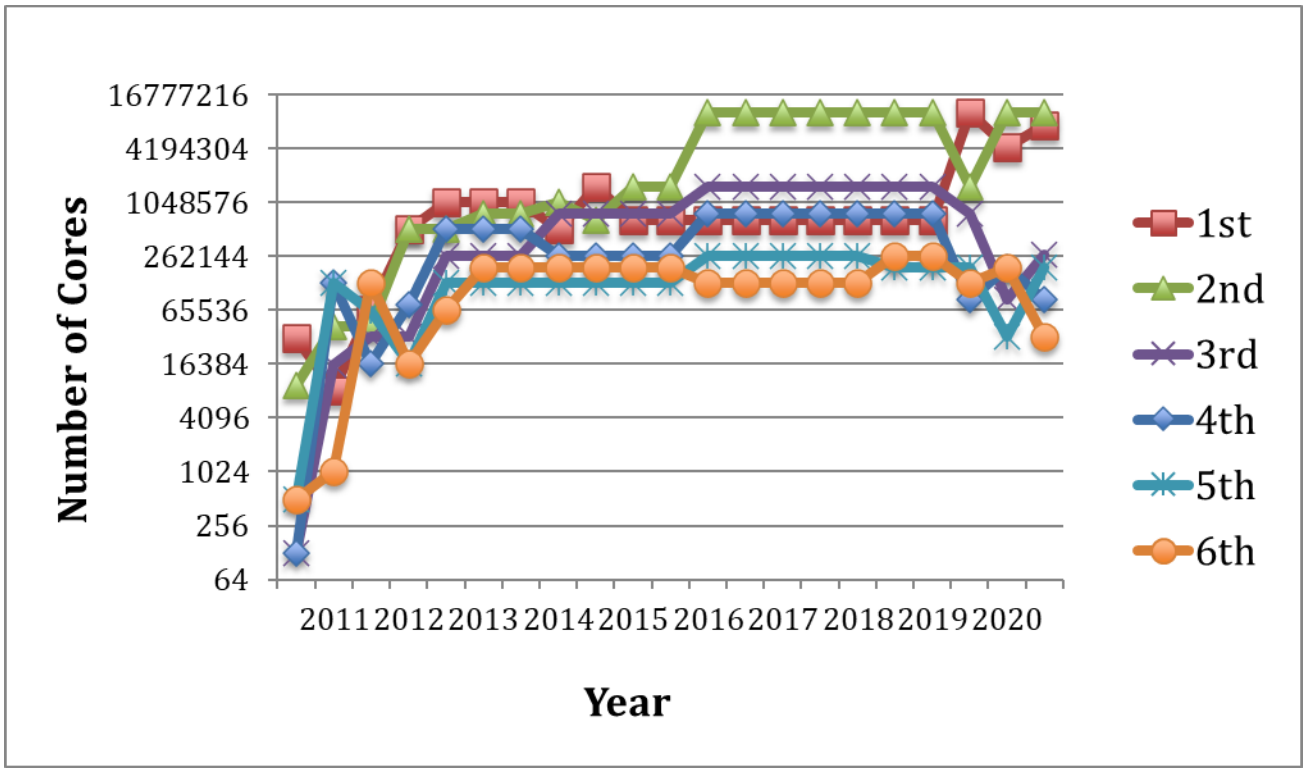


Figure AB-2 Number of Cores Used in the Top Six Graph 500 Machines Since the Inception of the Graph 500 Competition [data via graph500.org]

5.1.2. TECHNOLOGY NEEDS, SYSTEMS AND ARCHITECTURES IMPACT

The gains in graph processing performance come from three main sources: improvements in algorithms, increase in memory bandwidth, and increase in processor/memory interconnection network bandwidth. It is important to note that processor performance does not have a first-order impact.

Some of the gains in TEPS over the years have been due to improvements in the algorithm used. We expect the algorithms to keep changing. Programmers will attempt to exploit locality more, and to reduce the communication as much as possible, possibly through redundant recomputation. While we expect better algorithms to continue to have an impact, the improvements are likely to provide diminishing returns.

The most critical resources at this point are the bandwidth and latency of both the memory and the global network. This is where the Fugaku supercomputer excels. Graph problems have a high ratio of communication to computation. Moreover, they rarely have much locality. As a result, there are frequent, non-local transfers of data between the memory and the cores, and among the cores. Such transfers follow an irregular pattern.

The first source of gains in GTEPs for Fugaku comes from an increase in memory bandwidth. The top machines before Fugaku had an aggregate memory bandwidth of about 5 petabytes per second. Fugaku with 158,976 multicores has a total aggregate memory bandwidth of 163 petabytes per second. This is thanks to 32GB of in-package HBM2 DRAM per multicore. This 3D stacked memory has a memory bandwidth of 1024GB/s per node.

The second source of GTEP gains for Fugaku comes from a flexible high-dimension global network that provides high bandwidth and low latency. The Tofu D network is a physical 6-D mesh/torus network (on which a virtual 3-D torus is mapped as seen by the programmer). What is especially powerful is that it provides a very high injection bandwidth into the network: 40.8GB/s per node or 6.49 PB/s in total. In addition, it provides fast all-to-all communication support, thanks to the fact that, with the virtual 3-D torus, each node has six adjacent nodes and, therefore, topology-aware algorithms can use six simultaneous communications effectively. Of course, the cost of the network is very high. It will be interesting to see if future optical networks can provide a cost- and bandwidth-competitive solution.

One trend we are observing is that the graph sizes continue to increase. Since graph data is typically confidential, we do not know the sizes of the databases used by large commercial data centers and government agencies. However, we expect the databases used to use hundreds of terabytes or petabytes soon. Moreover, these graphs are dynamic, meaning that they continuously change in time. To process these graphs, we need very large clusters.

Looking forward, we expect that Fugaku will maintain its lead for a while, and that top-ranked graph machines will tend to resemble Fugaku. Specifically, we expect to continue to see simple processors. For example, the Fugaku supercomputer uses custom-designed multicores that contain 48 ARM-based cores (A64FX) at 2.2GHz, for a total of 7M cores. Since there is a high rate of memory accesses that miss in the caches, these cores do not need to be very wide-issue. They do not need to be equipped with substantial floating-point hardware. They are frequently stalled due to reorder buffer or load/store queues being full. One interesting issue is whether graphics processing units (GPUs) will be used, as they become more tolerant of computation divergence. However, it may be necessary to change the graph algorithms, typically reducing the efficiency of the algorithms.

Furthermore, the memory provisioned in terms of bandwidth and size will be high. This means that memory will be a sizable fraction of the cost of the machine. Note that memory needs are large in a machine with in-DRAM graphs—such problems need hundreds of terabytes and even petabytes. One interesting question is what can be accomplished with the arrival of relatively fast non-volatile memory (NVM). With NVM, the amount of memory available will increase substantially. However, it may require re-writing the algorithms.

Finally, networks will likely resemble the one from Fugaku: custom designed, high dimension, and providing fast all-to-all and irregular communication patterns. They will largely determine both the cost and the performance of the machine. It is possible that optics-based networks will deliver the necessary capabilities. Other forms of network topology that handle non-local traffic effectively are possible.

References for Section 5.1

[1] http://graph500.org/

[2] http://green.graph500.org

[3] Lifeng Nai, Yinglong Xia, Ilie G. Tanase, Hyesoon Kim, and Ching-Yung Lin, “GraphBIG: Understanding Graph Computing in the Context of Industrial Solutions”, Supercomputing, 2015.

[4] Vasiliki Kalavri, Vladimir Vlassov, and Seif Haridi, “High-Level Programming Abstractions for Distributed Graph Processing”, arXiv:1607.02646v1, Jul 2016.

5.2. Artificial Intelligence

The application area of Artificial Intelligence (AI), including the specific domain of Machine Learning (ML), refers to a class of techniques that learn from data, in order to enable applications that can make meaningful decisions—such as classifications, predictions or recommendations. As such, these techniques do not need to be explicitly programmed. Over the last few years, advancements in the family of AI/ML techniques known as Deep Neural Networks (DNNs) have demonstrated superhuman accuracies at tasks such as computer vision, speech recognition and language translation.

There were three important ingredients behind the success of this paradigm. First was the ready availability of vast datasets of labelled examples for training. Second was the inherent but non-obvious scalability of DNNs, which are based on techniques known since the mid-1980s—supervised training using backpropagation and stochastic gradient descent. What we know now is that when these types of networks are made larger, with more neurons and more weights, and are trained with more data, they almost always perform better. Thus, one of the key ingredients to the recent success of AI/ML have been powerful computational platforms. Hardware that was already readily available, such as CPUs and GPUs, were critical for empirically proving that the DNN approach was in fact scalable, and for the first applications. But now we seek to extend upon the inherent scalability of these algorithms by expressly designing hardware for AI/ML—using a mix of CPU, GPU, FPGA, custom accelerator ASICs, and perhaps even more exotic hardware approaches—in order to address a growing set of commercially relevant applications.

The focus of this section is on Application Benchmarking of the AI/ML area. Our goal is to track progress in this field, to enable continued improvement of existing computational approaches (e.g., CPUs and GPUs), and to provide accurate performance targets for both evolutionary and revolutionary computing approaches for this application area. We attempt to accurately report the recent trend-lines in the performance metrics of interest—throughput, latency, accuracy, and energy efficiency—so that we can attempt to extrapolate the future evolution of these trends, at least into the near-future. Figure AB-3 shows the various choices that must be made when implementing an ML application, and thus illustrates the significant difficulty of quantifying improvements enabled by hardware, given the presence of so many other factors that can (and do) influence ML performance metrics.

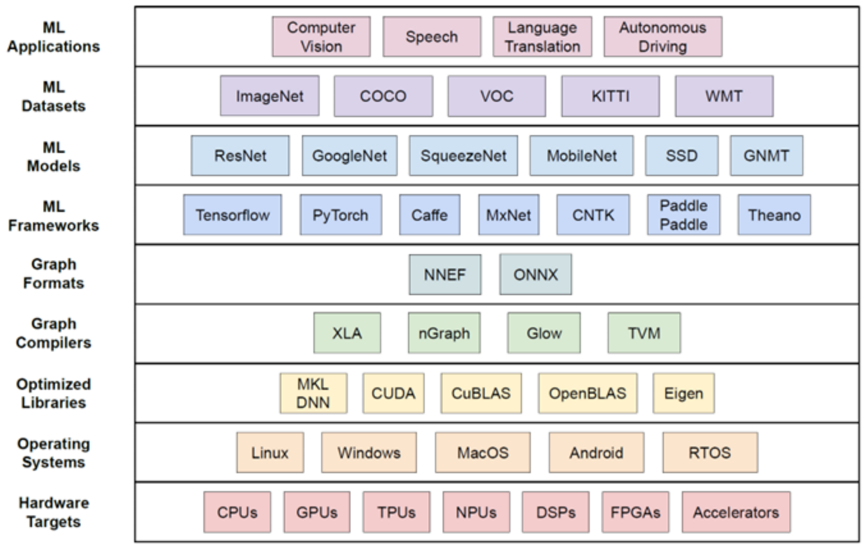


Figure AB-3 The variety of software and hardware options at various levels greatly complicate the benchmarking of AI/ML systems[1]

5.2.1. Application Domains

While AI/ML algorithms have been around for many years, the specific class of Deep Neural Networks have found commercial application on a wide variety of workloads in the past decade, including image classification, object detection, speech recognition, machine translation, recommendation systems, sentiment analysis/classification of text, language modeling, text-to-speech, face identification, image segmentation, and image enhancement[2]. In the business world, AI/ML is used for process automation, cognitive insight, fraud detection, cognitive engagement, and other applications. In consumers' interactions with devices like smartphones, and with large internet companies like Google, Facebook, YouTube, Netflix, Uber, Amazon and the like, it takes less time to list the functionality that has *no* dependence on AI/ML than it does to describe the functionality that now depends on AI/ML to some degree.

5.2.2. AI/ML Deployment Platforms

AI/ML applications are deployed on a wide variety of platforms, based on target application needs. The computational needs of these platforms vary based on application needs and platform form-factor. To understand the tradeoffs involved in choosing a specific AI/ML deployment mode, it is important to understand the different workload characteristics and their constraints.

Here we focus on the supervised learning behind Deep Neural Networks, for which AI/ML techniques have two different distinct phases – training and inference.

- Training is the process whereby labelled data is fed into the system and a trained model of a particular accuracy is built. The training of a model is not an absolute process, but can spit out models of varying accuracy, depending on how good the model is, on how well the data is chosen, and on careful tuning of various hyper-parameters. In general, more training data and larger models lead to higher accuracies, although at the cost of increasing the computation and memory requirements. As models become extremely large, training is more and more a distributed task, thus involving networking between multiple nodes all working together on the same overall task or set of tasks.
- Inference is the process whereby the trained model is deployed, actual input data is fed to it, and inferences are made.

Training and Inference need not run on the same hardware, and these compute tasks can be partitioned across multiple hardware platforms based on the target application constraints. The target application constraints are typically latency, power, cost, inference speed, training speed, and accuracy (or similar metric for decision quality).

For self-driving cars, latency is absolutely critical for the car to react to a pedestrian or a traffic signal to avoid accidents and so inference runs on an edge computer in the car. For a photo tagging application on a smartphone, there are no latency constraints, so cloud servers could be used for the inference. For a low-cost gas sensor that can sense dangerous gases using AI/ML methods, one can use inexpensive microcontrollers for running inference.

With this basic taxonomy, we describe the three basic deployment models based on the constraints shown in the table below: "High Performance" AI/ML performed in the cloud, "Real Time Inference" as performed at the Edge, and "Resource-constrained Inference." This last category is sometimes referred to as "Endpoint Inference" or "TinyML." It is also performed at the Edge, but by Micro-controller Units (MCU) or in Internet-Of-Things (IOT) sensors where resources such as cost, or the volume-, area-, or memory-footprint, or the amount of battery power/energy available may be severely constrained.

Training has high computational needs and so mostly runs on the cloud. The aforementioned strong correlation between the size of the training set and the model performance, and the difficulty of avoiding "catastrophic forgetting" when re-training a model with only some of the original training set[3], imply that there are significant challenges to an edge-based training approach. However, there is some research in running training on edge computers for data privacy reasons, with data scalability supported by using distributed or federated training methods that can exchange weight-update data and enjoy the benefits of network scale, without sacrificing data privacy or other data-locality requirements.

Table AB-4 AI/ML Deployment Platforms and Constraints

Deployment Platforms	***High Performance (Cloud)***	***Real Time Inference (IoT Edge)***	***Resource-constrained Inference (IoT Edge MCU/IOT)***
Training System	*Cloud*	*Cloud*	*Cloud*
Inference System	*Cloud*	*IoT Edge Computer*	*IoT Edge Microcontroller*
Training Speed	High	High	High
Inference Throughput (pages/second)	High	Medium	Low
Inference Latency (milliseconds)	High	Low	Low
Compute	High	Medium	Low
Cost	High	Medium	Low
Power	High	Medium	Low

5.2.3. Computing Architectures

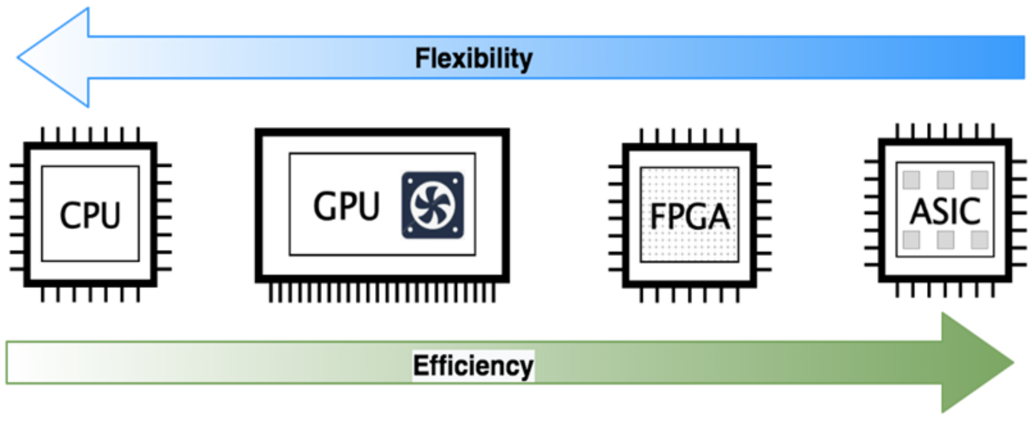


Figure AB-4 General tradeoffs among Computing Architectures

5.2.4. AI/ML Hardware

The AI/ML-specific portion of the market for computing hardware is growing rapidly, and is expected to account for almost 20 percent of all computing demand by 2025. This would translate into about $67 billion in revenue[4]. As mentioned previously, AI/ML applications run on a variety of platforms. Here we describe the hardware used across these platforms.

5.2.4.1. CPU

CPUs remain the ubiquitous choice for running AI/ML workloads. Inference is run heavily over many-cores in datacenters and on simple CPUs on smartphones or IoT devices. However, the end of performance scaling in CPUs means that getting increased performance as technology scales can no longer be expected. This becomes exceptionally challenging as DNN models are evolving rapidly.

5.2.4.2. GPU

GPUs were the hardware platform that propelled the current strong interest in AI/ML, since their parallel processing capabilities can be leveraged to accelerate matrix multiplications, which are at the heart of the computations within Deep Neural Networks. A single modern GPU can have upwards of 5,000 tiny cores capable of highly parallel operation. These systems also provide high internal memory bandwidth, which makes them ideal for the intense computation needs of AI/ML. NVIDIA's GPUs offer "Tensor Cores," tiny engines for running dense matrix multiplications efficiently at reduced precision. Training requires a high number of high-precision floating point operations with high memory bandwidth, and GPUs remain the most popular choice. For inference however, the large power envelope of most GPUs makes them less efficient for mass deployment in edge applications.

5.2.4.3. FPGA

FPGAs are a popular choice for DNN inference, as their reconfigurable substrate helps tailor the datapath for the DNN model. Moreover, FPGAs are bit-configurable, and are thus highly popular for running inference at lower precision for compute- and memory-efficiency. FPGAs can also offer field re-programmability as DNN models evolve. Today's FPGAs are less suited for training though, as they lack high-precision floating point units. The most vibrant example of FPGAs being used for AI/ML today is Microsoft's Brainwave project[2]. On the downside, the overall energy-efficiency and operating frequency of FPGAs is still lower than ASIC counterparts.

5.2.4.4. Digital ASIC Accelerators

The inefficiencies in CPUs, GPUs and FPGAs described above has led to a surge in specialized Digital ASIC-based accelerators for AI/ML[5]. From a high-level view, all digital DNN accelerators are essentially large arrays of Multiply-Accumulate (MAC) units fed via custom scratchpads, along with special functional units for pooling, activation, and so on.

There are two key aspects that can be used to classify the various prototype (commercial and academic) digital accelerators that have been demonstrated: ***microarchitecture*** and ***dataflow***.

Microarchitecture**—**At the microarchitecture level, we can classify these digital ASIC Accelerators into two categories, depending on the control paradigm of the MAC units: systolic array and SIMD/MIMD arrays.

Systolic arrays are built as a grid of MAC units connected as a pipeline in two dimensions. Data (inputs/weights/partial-sums) move every cycle in a horizontal, vertical (or both) direction, depending on the dataflow strategy. The entire array operates as a monolithic matrix-matrix multiplication core with a centralized controller. Individual MAC units cannot be stalled.

In contrast, SIMD/MIMD arrays are built using a collection of processing elements (PEs), with each containing a MAC unit and some local storage. These arrays can be operated as full MIMD units (i.e., every PE has independent control and can stall if it is waiting for operands), or multiple SIMD units, where the PEs within each SIMD unit operate in lock-step, but different SIMD units can be controlled independently.

Both approaches appear in products today. For instance, Google's TPU[6] uses a systolic array, while NVDLA (Nvidia Deep Learning Accelerator)[7] and TESLA FSD[8] use SIMD/MIMD spatial arrays. Systolic arrays are simpler and offer higher compute density but may suffer from under-utilization if the problem dimension does not map well over the array, or has sparsity (i.e., zeros). SIMD/MIMD arrays are better for utilization and provide more mapping flexibility but come at the cost of more complexity due to distributed control[9].

Dataflow**—**In this digital ASIC accelerator domain, dataflow collectively refers to the data iteration order and the partitioning strategy across the MAC units.

The choice of dataflow determines overall data reuse, which in turn affects the amount of data movement in/out of the array. Data reuse can be augmented in two ways: by reuse of weights and input excitations across the multiple data examples within a minibatch that is processed together, and by using networks with Convolutional Layers, in which the small number of weights in a convolution kernel are re-used across many input excitations, as the convolution kernel passes over an input or intermediate image. Weight reuse is particularly important, since it amortizes the large energy cost of retrieving data from memory (particularly from off-chip memory) over a much larger number of computations that reuse that data.

In terms of iteration order, accelerators can follow different strategies, including weight/input stationary (where the weight/input tensor is kept stationary and the input/weight tensor is streamed in) or output stationary (where the weight and input are both streamed in and the output is computed in-place). Google's TPU uses the weight-stationary approach[9].

For the partitioning strategy, different dimensions of the input/weight tensors can be partitioned across the MAC units for parallelism. For example, NVIDIA's NVDLA partitions the input channels along one dimension, and output channels along the other. For each dataflow, compilers can come up with appropriate tile sizes depending on the amount of on-chip buffering.

Some of these architecture and dataflow concepts being explored in the context of digital ASIC accelerators can be exploited in other contexts, particularly when programming FPGAs for these AI/ML applications. In addition, newer FPGA designs are starting to merge digital ASIC accelerator concepts within conventional FPGA configurations[10].

5.2.4.5. Compute-In-Memory (CIM) macros

Compute-in-Memory is a paradigm that achieves the parallel multiply-accumulate capabilities of the "Analog AI" techniques described below, but without requiring analog conductance states. Instead, digital memory devices such as SRAM or a compact NVM such as RRAM are used to encode multi-bit synapses in a customized memory array, which is then designed to perform some or all of the multiply-accumulate operations needed for AI/ML inference (and training) during the actual memory read[11]. These approaches typically involve multiple bit- or word-lines for each synaptic weight, frequently assuming factor-of-2 contributions from different bit- (word-)lines in such a way that the multiplication of the synaptic weight by the neuron excitation gets performed during the memory read. As such, these CIM macros depend more critically on tight device-to-device variability than conventional memory, but with the advantage of avoiding the time of accessing memory words individually, and the area and power of the circuit blocks that would otherwise be required to multiply and add the data. The advantages of SRAM-based CIM macros are that the high-endurance of SRAM allows small macros to be reused across many different weight values, so that macro size does not need to match the total size of the AI/ML model; the advantages of RRAM-based CIM macros are, at least eventually, in higher density (e.g., lower area-per-bit).

5.2.4.6. Analog AI

An emerging class of accelerators is based on analog computing—leveraging current summation within dense analog arrays to perform large vector-matrix computations extremely efficiently, allowing both high speed and high energy efficiency[12]. This approach is particularly well-suited to DNN layers with small weight-reuse factors, such as the fully-connected layers found in LSTM and Transformer networks. However, many of these analog systems depend on emerging nonvolatile memory devices, which are not yet a widely-available technology in semiconductor foundries, or on the extension of existing nonvolatile memories that have stopped scaling, such as NOR-FLASH. In addition, it is not yet clear whether the inherently limited signal-to-noise ratios in analog systems can be brought high enough to robustly support the very large models used in modern DNN systems. In contrast, the designer of a digital accelerator can choose to dial down the precision of the digital computations almost arbitrarily, aiming for the best possible energy efficiency while still fully supporting the original neural network accuracy.

5.2.5. Considerations for optimization

In this section, we list some of the design-knobs being used extensively today to gain performance while minimizing any loss in accuracy/precision, as well as important variables that must be considered, ranging from software through mixed hardware-software to fully hardware-specific techniques.

5.2.5.1. Quantization

Quantization is the idea of running inference at lower precision[13]. Most modern DNNs perform inference at 16b and 8b, and in fact some also go lower to 4b. Lower precision naturally reduces the amount of data to move, the time to perform a multiply-accumulate, and add more compute resources in the available area. Both the total number of bits, and for floating-point formats, the number of bits dedicated to the exponent, can have a significant impact.

5.2.5.2. Pruning

Pruning is an approach leveraged during training, whereby certain excitations, weights, or both are set to zero based on some heuristic (e.g., threshold-based, or block-based)[14]. It has been shown that pruned networks can operate at the same accuracy as the dense ones, if additional training is performed after pruning. Pruning can decrease the required memory footprint by reducing the total number of weights, as well as decrease total runtime and latency (and increase throughput) by reducing the required number of computations. However, the subsequent DNN is then described by sparse matrices. Unless the hardware is expressly designed to take advantage of sparsity, this can affect the utilization of the compute units within hardware.

5.2.5.3. Hardware-software co-design

Hardware-software co-design is an approach where the hardware AND the neural network models are jointly designed together. This path can lead to significantly higher energy efficiencies, although it works best when the set of DNN models that must be supported is small (typically a small set of Convolutional Networks for particular image-processing applications), and when the long cycle time for custom hardware can be tolerated.

5.2.5.4. Utilization

Utilization is the fraction of time that digital processing elements—within a CPU, GPU or digital accelerator—are performing useful computations. Typically, poorly-designed dataflows can lead processing elements to be idle because the next data to be computed has not yet arrived. Given the large amounts of leakage power in the many transistors within digital systems, keeping utilization high by minimizing any idle processing elements is key for high energy efficiency.

5.2.6. Benchmarking

A standardized and robust benchmarking methodology is crucial, both for making the right tradeoffs across this varied hardware landscape during platform selection, and for identifying key bottlenecks that in turn can drive innovation in the ecosystem.

The metrics of interest when benchmarking AI platforms depend on the use-case of inference vs training. Table AB-4 highlighted the differences in platform capabilities depending on the deployment model.

5.2.7. Training in the Cloud

For training in the cloud, the metrics of interest are typically the final trained accuracy, and the runtime needed to get to that specific accuracy target. While accuracy is fundamentally dependent on the DNN model and its datasets, it also depends on the parallelization strategy and hardware precision employed. For instance, data parallel training divides the training set into "mini-batches"—and the size of the mini-batch has been found to affect overall training accuracy. Training with 32-bit floating point, vs 16-bit floating point formats (e.g., including the custom bfloat16 format, which offers the same range of exponents as the 32-bit format) can also affect overall accuracy. The runtime to get to a specific accuracy target depends on the efficiency of the hardware platform, which depends on both the capability of a single node (e.g., GPU) and the memory plus network capabilities of a cluster of nodes (when running distributed training). Other approaches for speeding distributed training involve careful compression of the weight update data exchanged between the nodes.

5.2.8. Inference in the Cloud

Inference in the cloud is characterized by end-to-end latency, and the observed accuracy/quality of the output. Inference on the cloud often leverages mini-batching as well, which increases energy efficiency by data reuse. However, given the stochastic nature of AI/ML inference requests, latency constraints can sometimes force the sub-optimal execution of inference jobs with less than a full mini-batch of input data.

5.2.9. Inference at the Edge

Inference at the edge is characterized by real-time latency, the observed accuracy/quality of the output, and energy-efficiency. This is because edge platforms are highly energy constrained, and trading off accuracy for runtime and energy-efficiency is often an acceptable trade-off. There is thus an emerging trend in developing edge-platform friendly DNN models (e.g., MobileNetv2, SqueezeNet) leveraging frameworks such as TensorFlowLite. Typically, edge inference is performed in smaller batches, although inference in autonomous vehicle may involve a continuous stream of images from multiple sensors.

5.2.10. Training at the Edge

As discussed in Section 4.2.2, training on the edge is another possible deployment platform, although not yet mainstream. Here the metrics of interest might be the amount of new data needed to improve the accuracy of the model while under deployment.

5.2.10.1. Benchmarking Environment

For an effective AI/ML benchmark, first the benchmark would have to standardize the testing/benchmarking environment. For AI/ML applications, this involves standardizing models based on use-cases like computer vision, speech recognition and others. It is evident that given the difference in processing, the benchmark should be separate for Training and Inference processes. Some of the recommended metrics that can be captured as part of the benchmarking efforts are listed in Table AB-5.

Table AB-5 AI/ML Benchmarking Metrics

	Description	**Units**
Compute	Raw Compute Capability	TOPS (Tera, or 1e12 operations per second)
Inference Speed	Number of features classified per second	Frames/Sec
Training Speed	Time to train any model at specified accuracy, e.g., 90% accuracy	Time in minutes
Power	Processing output / Watt	Inference Speed/Watt Training Speed/Watt

5.2.10.2. MLPERF.ORG

In May 2018, a consortium of companies and universities announced[15] the creation of the MLperf.org website and github repository. Currently grown to 64 companies and 8 universities, the consortium's website describes two broad benchmark suites for ML applications, in both training and inference.

Motivated by the SPEC benchmark, "MLPerf's mission is to build fair and useful benchmarks for measuring training and inference performance of ML hardware, software, and services. We believe that a widely accepted benchmark suite will benefit the entire community, including researchers, developers, hardware manufacturers, builders of machine learning frameworks, cloud service providers, application providers, and end users.

"Our goals include:

- Accelerate progress in ML via fair and useful measurement
- Serve both the commercial and research communities
- Enable fair comparison of competing systems yet encourage innovation to improve the state-of-the-art of ML
- Enforce replicability to ensure reliable results
- Keep benchmarking effort affordable so all can participate"[16]

Recently, the first set of MLperf Inference results (v0.5) were released[2]. Other AI/ML benchmarks include AI Benchmark, EEMBC MLMark, Fathom, AIXPRT, AI Matrix, DeepBench, TBD (Training Benchmarks for DNNs), and DawnBench. [2 and references within]. So far, no MLperf results have included power, although this is planned for future versions of the benchmark.

The goals of the MLperf organization are very closely aligned with this chapter. Key differences are that MLperf focusses on a current "snapshot" of AI/ML performance, and explicitly compares particular systems with each other by name. Improvements that arise from better frameworks, microcode, or mapping from frameworks may be conflated with improvements coming from the underlying hardware. In contrast, the IRDS roadmap is more interested in longer-term trends in improved AI/ML systems, as explicitly traceable to changes in the underlying hardware.

If MLperf is the right source for information on which commercial system to acquire today, the goal we are striving towards with this IRDS/AB section is to predict what level of AI/ML performance will be commercially relevant in 2–3 years' time.

5.2.10.3. DIFFICULTIES IN BENCHMARKING

While this is a lofty goal, there is well-known Danish saying, sometimes attributed to Niels Bohr: "It is difficult to make predictions, especially about the future."

A key challenge with designing and benchmarking AI platforms is the tightly integrated stack—from the DNN models down to hardware (Figure AB-3). Unlike traditional SW-HW platforms where clear abstractions exist, the efficiency of AI platforms actually comes from breaking the SW-HW boundary. This makes it challenging to tease apart benefits coming from innovations in hardware, compilers, and DNN development frameworks (e.g, TensorFlow, PyTorch).

While there are some emerging benchmarks that focus on essential component blocks of DNN systems[17], it is not clear that a benchmarking approach focused solely on component blocks could accurately track full forward-evaluation or training times, nor how such an approach would provide reassurance that an approximate digital or analog technique was achieving effectively identical classification accuracies.

Finally, compute efficiency is typically measured in terms of TOPS/W, short for TeraOPS/W, or 1e12 operations per second per Watt. Older hardware is sometimes rated in terms of TFLOPS/W, but the use of reduced precision has led to the need to exclude the phrase "Floating Point" to avoid confusion. In terms of energy efficiency, there is a significant difference between peak and actual compute efficiency, especially for DNN models that fall well below 100% utilization. For digital accelerators, this can be quantified nicely in the form of a rooftop curve, plotting computational performance as a function of operations[9].

5.2.11. PERFORMANCE TRENDS AND PREDICTION

In the meantime, here we show some publicly available data on raw computational capabilities of various systems either released or announced for training and inference of DNNs.

Figure AB-5 shows a plot of computational efficiency in units of TOPS/W for recent CPUs, GPUs, FPGAs and projected or advertised capabilities for some custom DNN hardware (labelled as ASICs). Here we focus on hardware that could be used for training, using either FP32 or BF16 precision formats. For completeness, we also include FP16 hardware as well, although some reports indicate that not all models can be trained accurately with the original half-precision format without adaptations, and that the BF16 format—which uses the same number of bits for the exponent as FP32, but many fewer in the mantissa—is required. On the other hand, there are also recent reports that training can be done at 8-bit precision as well[18]. Figure AB-6 replots the data from Figure AB-5, showing how TOPS (data points) and the resulting TOPS/W (assessed against the green dotted lines) vary as a function of reported system power.

The separate spreadsheet associated with this report shows a breakdown of numbers used to generate this plot, including references for where each data point was obtained.

Figure AB-7 shows a similar plot of computational efficiency in units of TeraOPS/W for numerous recent CPUs, GPUs, and FPGAs, as well as projected or advertised capabilities for some near-future DNN hardware explicitly designed for inference and labelled as ASICs. Note that across these plots, one-time improvements based on reduced precision (from 32 to 16 to 8 bits) are clearly revealed. Figure AB-8 analogously breaks out the TOPS and TOPS/W performance as a function of reported system power.

Finally, Figure AB-9 plots the efficiency of AI/ML compute as a function of claimed compute performance in TOPS. Efficiency is computed by taking the ratio of actual TOPS delivered in EEMBC testing on three workloads (SSDMobileNet 1.0, MobileNet 1.0, and ResNet-50 1.0)[16] to the claimed peak TOPS. Details of these calculations, including references for the performance results, are available in the separate spreadsheet associated with this report. Even for well-known convolutional neural networks that feature significant weight reuse, and which are widely known and used, only a portion of the claimed peak TOPS are apparently available when running on real workloads. For higher peak TOPS, which are also associated with higher power systems, efficiency can be at or below 10%.

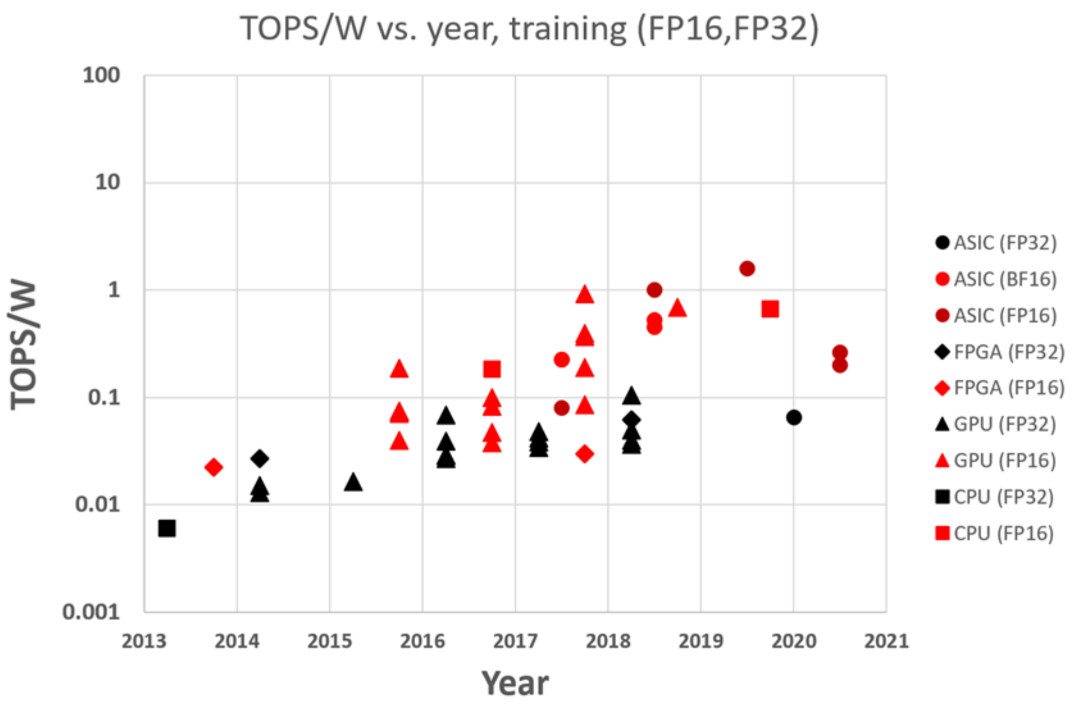


Figure AB-5 Compute Efficiency (TOPS/W) of Existing and Projected Hardware for Training

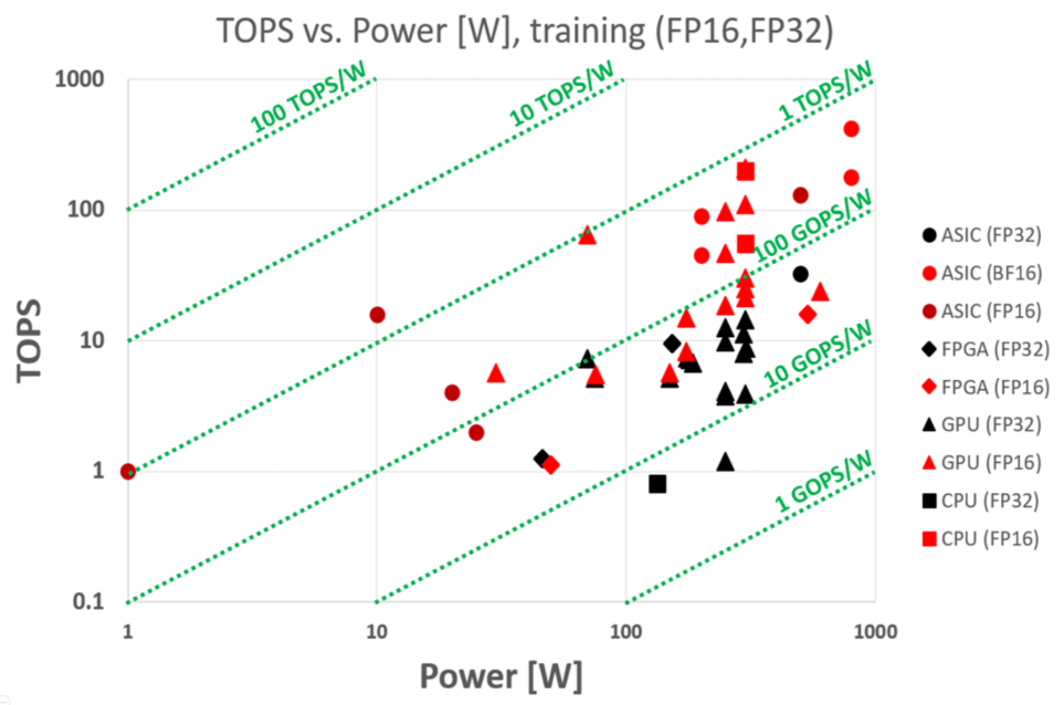


Figure AB-6 *Data from Figure AB-5 for Existing and Projected Hardware for Training replotted as TOPS as a function of reported system power*

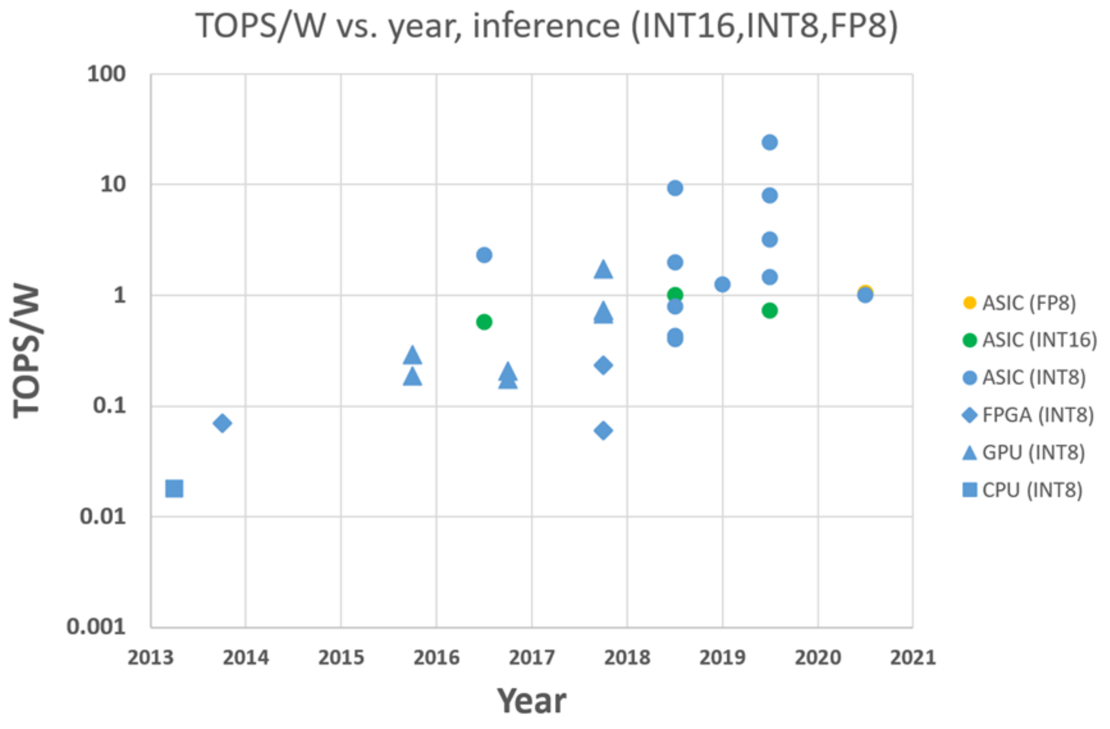


Figure AB-7 *Compute Efficiency of Existing and Projected Hardware for Inference*

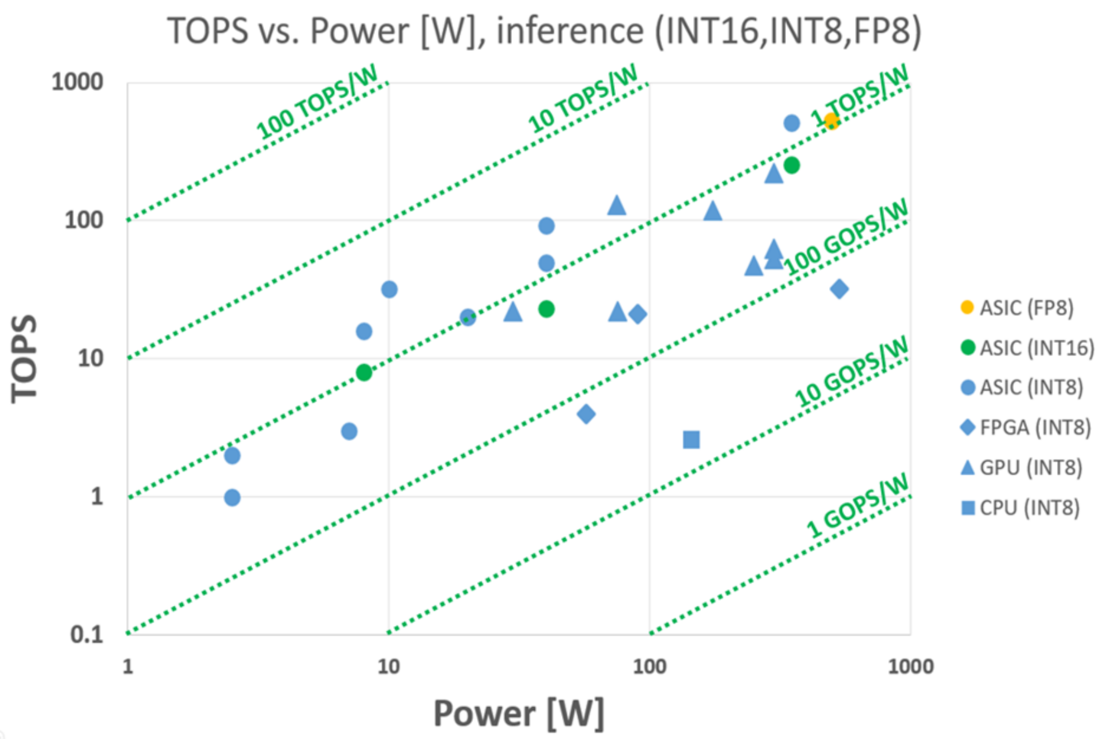


Figure AB-8 Data from Figure AB-7 for Existing and Projected Hardware for Inference replotted as TOPS as a function of reported system power

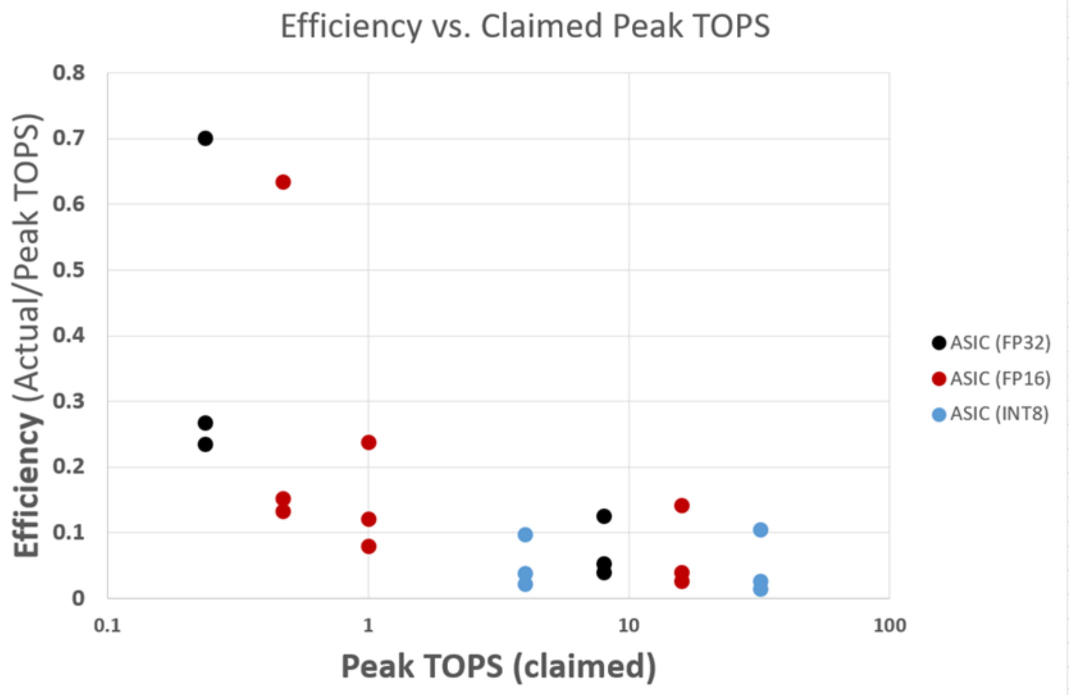


Figure AB-9 Efficiency as a function of claimed peak TOPS, showing the percentage of claimed compute power actually delivered in practice

5.2.12. Technology Needs, System and Architectures Impact

Key technology needs for Artificial Intelligence depend on the computing architecture (see Table AB-4). Generally, efficiency increases with specialization. Thus, more specialized FPGA and ASIC approaches for both training and inferencing will lead to the highest efficiencies. Any technology that enables higher efficiencies in FPGA and ASIC implementations will accelerate Artificial Intelligence. Also technologies that enhance near memory computation will improve pruned DNNs since they in turn result in sparse matrices.

Approaches to deployment platforms and their corresponding constraints are summarized in Table AB-4. As can be seen, the highest performance deployment platforms are the least energy constrained (e.g., cloud platforms). AI remains constrained by the memory bandwidth and latency. Because of optimizations such as pruning's impact on memory access patterns due to sparsity, increasing the capacity of caching does not necessarily lead to higher performance. Training is constrained also by the available parallelism on more commodity platforms such as multicore processors and GPUs.

References for Section 5.2

[1] V. Janapa Reddi, C. Cheng, D. Kanter, P. Mattson, et al., "MLperf Inference Benchmark," https://arxiv.org/abs/1911.02549
[2] https://www.microsoft.com/en-us/research/project/project-brainwave/
[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
[4] https://www.mckinsey.com/industries/semiconductors/our-insights/artificial-intelligence-hardware-new-opportunities-for-semiconductor-companies
[5] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE 105(12), 2295-2329 (2017).
[6] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates et al., "In-datacenter performance analysis of a tensor processing unit." In 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1-12. IEEE, 2017. https://arxiv.org/abs/1704.04760
[7] http://nvdla.org
[8] D. Das Sarma and G. Venkataramanan, "Compute and redundancy solution for Tesla's Full Self driving computer," Hot Chips 2019. www.hotchips.org ; https://fuse.wikichip.org/news/2707/inside-teslas-neural-processor-in-the-fsd-chip/
[9] https://www.sigarch.org/dnn-accelerator-architecture-simd-or-systolic/
[10] https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf
[11] N. Verma, H. Jia, H. Valavi, et al., "In-Memory Computing: Advances and prospects." IEEE Solid-State Circuits Magazine, 11(3), 43-55 (2019).
[12] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing." Proceedings of the IEEE 107(1), 108-122 (2018).
[13] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision." In Proc ICML, pp. 1737–1746. (2015).
[14] S. Han, H. Mao, and W. J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv:1510.00149 (2015).
[15] https://www.hpcwire.com/2018/05/02/mlperf-will-new-machine-learning-benchmark-help-propel-ai-forward/
[16] http://mlperf.org
[17] https://svail.github.io/DeepBench/
[18] N. Wang, J. Choi, D. Brand, C.-Y. Chen and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in 32nd Conference on Neural Information Processing Systems, 2018.

5.3. Physical System Simulation

Computer simulation of physical real-world phenomena emerged with the invention of electronic digital computing and is increasingly being adopted as one of the most successful modern methods for scientific discovery. One of the main reasons for this success has been the rapid development of novel computer technologies that has led to the creation of powerful supercomputers; large distributed systems such as high-performance computing facilities; access to huge data sets, and high throughput communications. In addition, unique and sophisticated scientific instruments, and facilities, such as giant electronic microscopes, nuclear physics accelerators, or sophisticated equipment for medical imaging are becoming integral parts of those complex computing infrastructures. Subsequently, the term 'e-science' was quickly adopted by the professional community to capture these new revolutionary methods for scientific discovery via computer simulations of physical systems[1].

The most important application codes for physical system simulations are typically based on discretization of partial differential equations using finite-difference, finite-element, boundary element, and spectral-element methods. Another form of mesh-less discretization leads to particle-based methods. Both types of computation constitute heavy workloads that were conventionally dominated by floating-point arithmetic, while an increasing number of applications are becoming increasingly bottlenecked by the memory bandwidth. Example applications include areas such as aerospace, astrophysics, biological and medical systems, climate and weather, combustion, materials science, and fusion engineering. Physical system simulation is now an essential tool in almost every branch of science and engineering where experiments are impossible, dangerous, or inordinately costly.

From the point of view of application programmers and end-users, the following major benchmarking efforts have been part of the development of this field over the years:

a. The NAS Parallel Benchmarks (NPB) include the descriptions of several (initially eight) "pencil and paper" algorithms[2]. All are realistic kernels, although the authors' claim that they include three "simulated applications" as a more accurate description 30 years ago than it is today. The NPB cover only the Computational Fluid Dynamics (CFD) application domain that is a primary interest of NASA.

b. The GENESIS Distributed-Memory Benchmarks[3] were developed in a 3-layer hierarchy – low-level micro-benchmarks, kernels, and compact applications. This was intended to express the performance of higher-level codes via a composition of performance results produced by the codes in the layer below. However, this proved to be a difficult task, particularly with including sufficiently broad computational science codes in the compact application layer.

c. The PARKBENCH Public International Benchmarks for Parallel Computers[4]. This was an ambitious international effort to glue together the most popular parallel benchmarks at that time – NPB, GENESIS, and several kernels including LINPACK. The PARKBENCH suite adopted the hierarchical approach from GENESIS together, thus inheriting the same difficulties described above.

d. All major machine vendors have participated in the development of SPEComp2001, since achieving portability across all involved platforms was an important concern in the development process[5]. The goal was to achieve both functional and performance portability. Functional portability ensured that the makefiles and run tools worked properly on all systems, and that the benchmarks ran and validated consistently. To achieve performance portability, SPEComp2001 accommodated several requests by individual participants to add small code modifications that took advantage of key features of their machines. There are many SPEComp2001 benchmarking results available, but their main role is to confirm that new hardware products and platforms have been validated by the vendors.

e. Another more recent "pencil and paper" parallel benchmark suite is the Dwarfs Mine based on the initial "Seven Dwarfs" proposal (2004) by Phillip Colella. The Dwarfs (computation and communication patterns) are described as well-defined targets from algorithmic, software, and architecture standpoints. The number of Dwarfs (which are really kernels with some of them mapped to NPB) was then extended to 13 in the "View from Berkeley" Technical Report[6]. The report confirms "presence" of the 13 Dwarfs in 6 broad application domains – embedded computing, general-purpose computing, machine learning, graphics/games, databases, and RMS (recognition/mining/synthesis). While this is a very interesting approach, the availability of results is very limited and even more importantly, the application domains are different from the ones selected by IRDS AB IFT. Some recent studies suggest that more Dwarfs (kernels) should be added for other application domains, while it is also not clear if the existing ones are sufficient for the domains described in the "View from Berkeley" Technical Report.

5.3.1. *Performance Trends and Prediction*

Most of the benchmarking projects mentioned above cover predominantly legacy dense applications, in which high computational intensity carries over when parallel implementations are built to solve bigger problems faster than can be handled by single processing server nodes. This means that inter-node communication can carry far less data per second than memories provide inside a node, and communication over such interconnection networks can be supported largely by explicit program calls to software stacks that manage the data transfers. As long as emphasis was on dense problems, this approach resulted in systems with increasing computational performance and was the presumption behind the selection of the LINPACK benchmark for the very popular semi-annual TOP500 rankings of supercomputers[7].

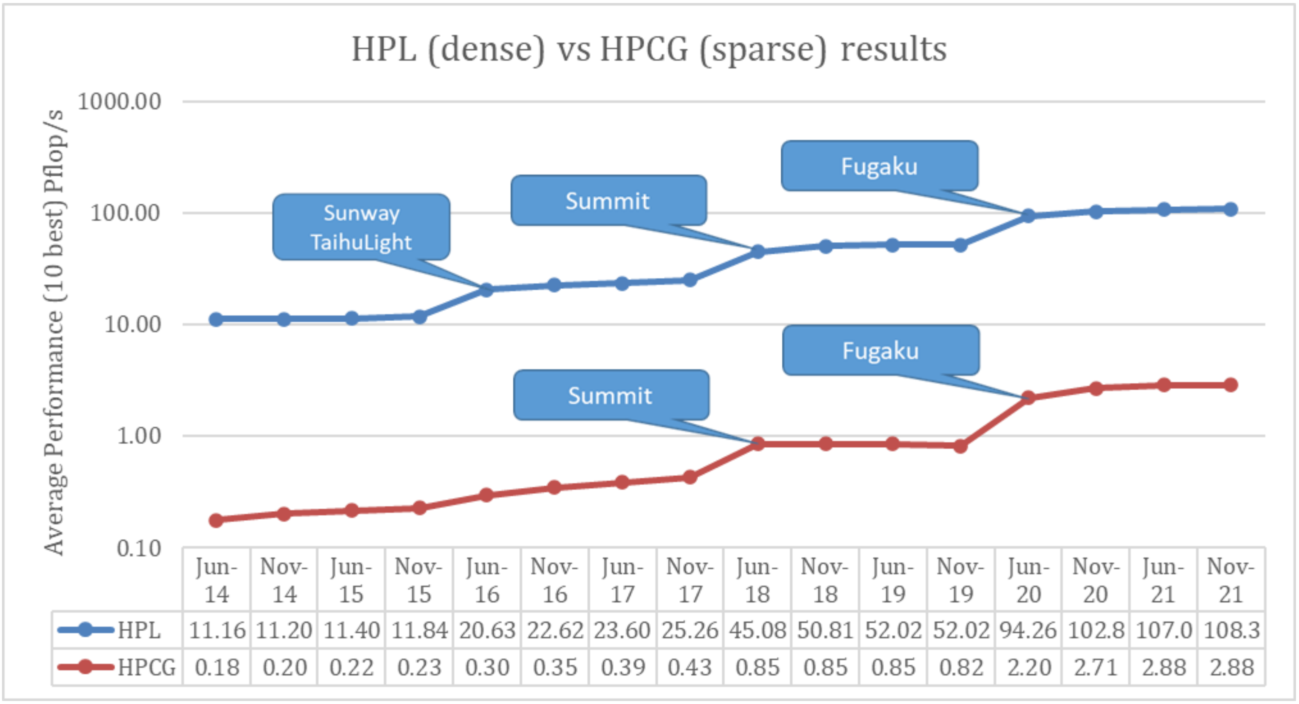
However, many new applications with very high economic potential—such as big data analytics, machine learning, real-time feature recognition, recommendation systems, and even physical simulations - have been emerging in the last 10-15 years. These codes typically feature irregular or dynamic solution grids and spend much more of their computation in non-floating-point operations such as address computations and comparisons, with addresses that are no longer regular or cache-friendly. The computational intensity of such programs is far less than for dense kernels, and the result is that for many real codes today, even those in traditional scientific cases, the efficiency of the floating-point units that have become the focal point of modern core architectures has dropped from the >90% to <5%. This emergence of applications with data-intensive characteristics—e.g., with execution times dominated by data access and data movement—has been recognized recently as the 3rd Locality Wall for advances in computer architecture[8].

To highlight the inefficiencies described above, and to identify architectures which may be more efficient, a new benchmark was introduced in 2014 called HPCG[9] (High Performance Conjugate Gradient). HPCG also solves Ax=b problems, but where A is a very sparse matrix—normally, with 27 non-zeros in rows that may be millions of elements in width. On current systems, floating point efficiency mirrors that seen in full scientific codes. For example, one of the fastest supercomputers in the world in terms of dense linear algebra is the Chinese TaihuLight, but that same supercomputer can achieve only 0.4% of its peak floating-point capability on the sparse HPCG benchmark. The current fastest supercomputer in the world in terms of LINPACK score is the Japanese Fugaku machine, but this supercomputer can also achieve only 2.3 % of its peak floating-point capability on HPCG. Detailed analysis lead to the conclusion that HPCG performance in terms of useful floating-point operations is dominated by memory bandwidth to the point that the number of cores and their floating-point capabilities are irrelevant[10]. There are of course application codes with highly irregular and latency-bound memory access that deliver significantly lower performance, but they are uncommon. While HPCG does not represent the worst-case scenario, it has been widely accepted as a typical performance yardstick for memory-bound applications.

In response to the increasing demand for low-precision arithmetic from the deep learning community, several vendors are now designing specialized hardware with fast low-precision arithmetic units. For the training of deep neural networks, a mix of 16 bit and 32 bit floating-point is commonplace, and for inference even lower precision can be used without detrimental effects. However, such low precision is not tolerable for most scientific applications but mixing them with higher precision can yield superior throughput while retaining sufficient accuracy. Many supercomputers in the Top500 are now equipped with NVIDIA GPUs that contain TensorCores, which perform matrix multiplication in mixed 32-bit/16-bit precision. The accuracy of computation on such hardware can be HPL further increased to double-precision though iterative refinement when solving a linear system of equations[11]. HPL-AI is a mixed-precision version of HPL, which utilizes iterative refinement to solve a linear system of equations up to the same accuracy as the original HPL but using much less time and energy.

Therefore, our selected benchmark codes that cover the “Physical System Simulation” application area of interest are the High-Performance LINPACK (HPL)[12][13], the HPCG[14], and the HPL-AI[15]. Both HPL and HPCG are very popular codes with very good regularity of results since June 2014, while HPL-AI is a new benchmark that has been tracked since November 2019. Another very important reason for selecting and HPCG is that they represent different types of real-world phenomena—the HPL models dense physical systems while the HPCG models sparse physical systems. Therefore, the available benchmarking results provide excellent opportunities for comparisons and interpretation, as well as laying out a relatively well-balanced overall picture of the whole application domain for physical system simulation.

Our approach for HPL and HPCG is to explore a 3-dimensional space—dense systems performance, sparse systems performance, and energy efficiency for both cases[16]. With HPL as the representative of dense system performance and HPCG as the representative for sparse systems, there are readily available performance and energy results published twice per year (June and November) with rankings of up to 500 systems for those two benchmarks since June 2014. We have further decided to use the average of the top 10 performance and energy results for each of these two benchmarks. This latter choice could be a point for further discussion and optimization of the benchmarking approach for this application domain. We have selected the 10 best only (rather than a larger number) because of the very limited HPCG results in the early years of publicly available HPCG measurements.

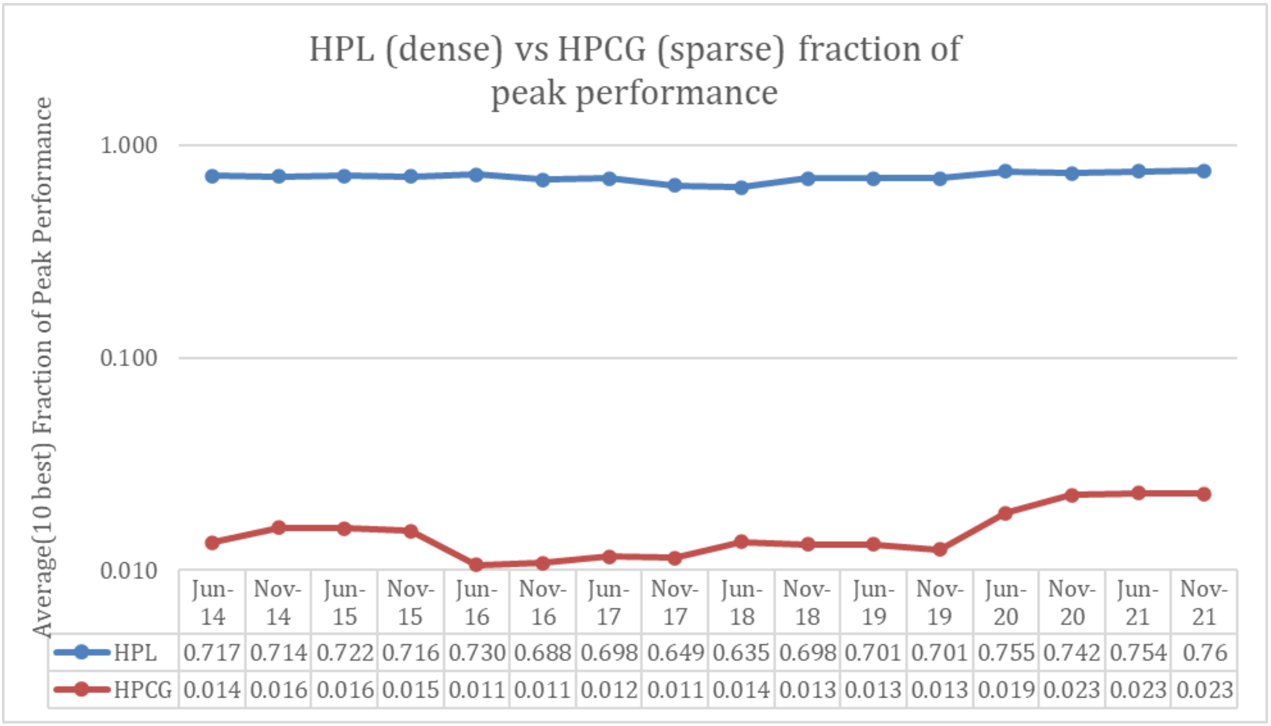


	Jun-14	Nov-14	Jun-15	Nov-15	Jun-16	Nov-16	Jun-17	Nov-17	Jun-18	Nov-18	Jun-19	Nov-19	Jun-20	Nov-20	Jun-21	Nov-21
HPL	11.16	11.20	11.40	11.84	20.63	22.62	23.60	25.26	45.08	50.81	52.02	52.02	94.26	102.8	107.0	108.3
HPCG	0.18	0.20	0.22	0.23	0.30	0.35	0.39	0.43	0.85	0.85	0.85	0.82	2.20	2.71	2.88	2.88

Figure AB-10 HPL (dense systems, blue) vs. HPCG (sparse systems, red) Average Performance

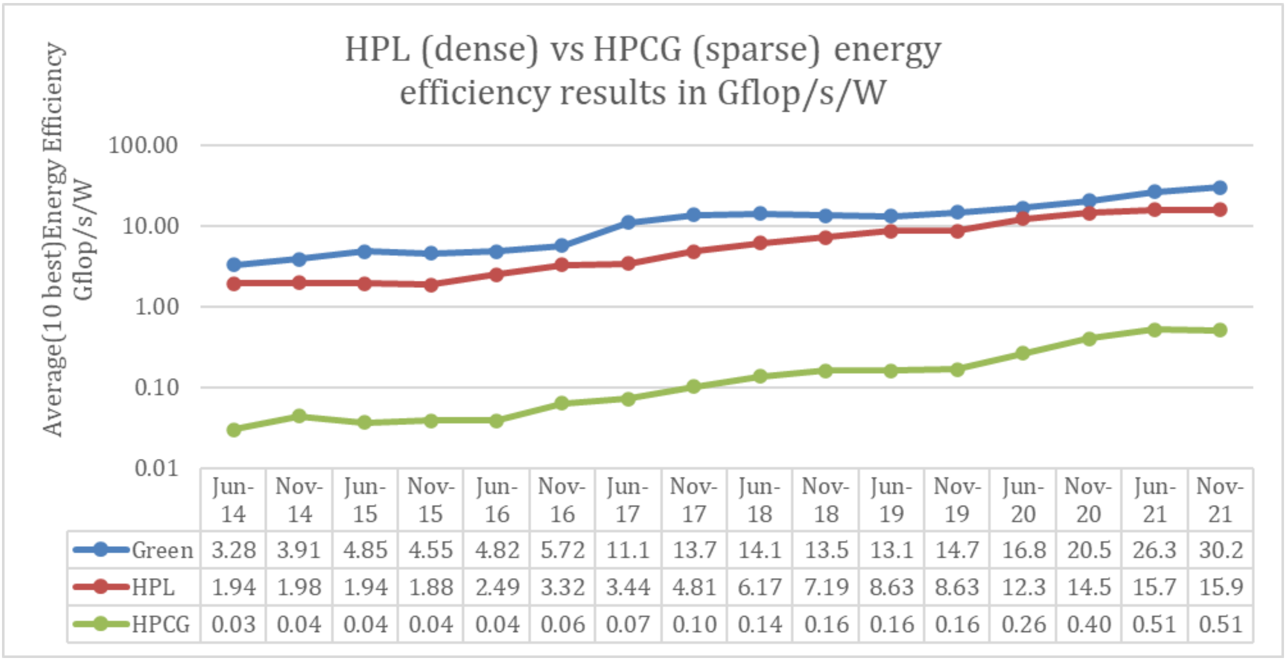
Figure AB-10 shows a significant performance gap of nearly 2 orders of magnitude between HPL and HPCG results in the last several years. The increase of the average HPL performance since June 2016 is because of the introduction of the Chinese Sunway TaihuLight system. The increase of both HPL and HPCG performance in June 2018 is due to the installation of the Summit supercomputer at ORNL. The most recent jump in 2020 for HPL and especially for HPCG is due to the installation of the Fugaku supercomputer at Riken, Japan. An optimistic expectation here would be to observe that the gap keeps closing and then assess the rate of this progress. Unfortunately, we do not have any evidence that the observed performance gap is in fact closing to any degree. Thus, we can draw the conclusion that one of the main challenges ahead will be to significantly increase sparse systems performance with any future computing systems designed for this application domain. While it is clear that reaching ExaFLOP/s performance with HPL will happen soon, it is equally clear that this achievement will leave this significant gap between dense and sparse systems performance unchanged.

Figure AB-11 complements the above analysis by showing a similar gap of approximately 2 orders of magnitude for the fraction of peak performance between HPL (dense) and HPCG (sparse). This provides clear evidence of something we have known for years – our production codes (which are usually sparse systems) are unable to deliver more than a few percent of the peak system performance that HPL results would seem to promise. The figure shows that this gap has not been reducing, and further points out the need to address sparse system performance in the next generation of computer architectures designed for this application domain.



	Jun-14	Nov-14	Jun-15	Nov-15	Jun-16	Nov-16	Jun-17	Nov-17	Jun-18	Nov-18	Jun-19	Nov-19	Jun-20	Nov-20	Jun-21	Nov-21
HPL	0.717	0.714	0.722	0.716	0.730	0.688	0.698	0.649	0.635	0.698	0.701	0.701	0.755	0.742	0.754	0.76
HPCG	0.014	0.016	0.016	0.015	0.011	0.011	0.012	0.011	0.014	0.013	0.013	0.013	0.019	0.023	0.023	0.023

Figure AB-11 HPL (dense systems, blue) vs. HPCG (sparse systems, red) Fraction of Peak *Performance*

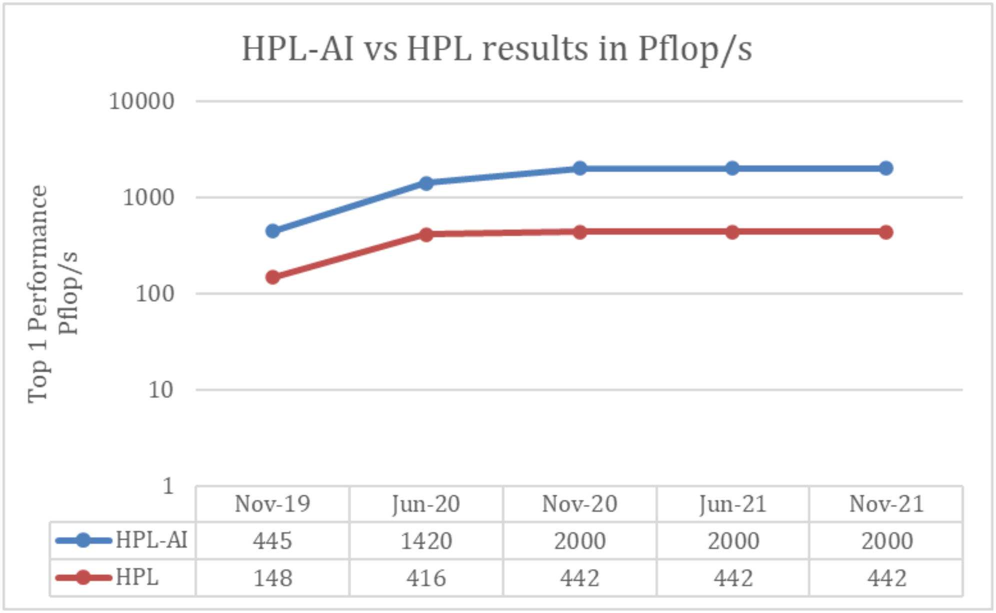


	Jun-14	Nov-14	Jun-15	Nov-15	Jun-16	Nov-16	Jun-17	Nov-17	Jun-18	Nov-18	Jun-19	Nov-19	Jun-20	Nov-20	Jun-21	Nov-21
Green	3.28	3.91	4.85	4.55	4.82	5.72	11.1	13.7	14.1	13.5	13.1	14.7	16.8	20.5	26.3	30.2
HPL	1.94	1.98	1.94	1.88	2.49	3.32	3.44	4.81	6.17	7.19	8.63	8.63	12.3	14.5	15.7	15.9
HPCG	0.03	0.04	0.04	0.04	0.04	0.06	0.07	0.10	0.14	0.16	0.16	0.16	0.26	0.40	0.51	0.51

Figure AB-12 *HPL (dense systems) vs. HPCG (sparse systems) vs. the most energy-efficient supercomputers on the Green 500 list*

The energy efficiency dimension of our evaluation is depicted in Figure AB-12. The current supercomputing designs appear to be able to scale up to 400 PetaFLOP/s while remaining within the recommended 20 MW system power consumption envelope. An optimistic estimate based on this would require 3.75 times improvements in energy efficiency, and 2.5 times improvements in the HPL performance currently delivered by the Fugaku supercomputer to reach the ExaFLOP/s barrier. Unfortunately, this would only achieve the desired performance and energy efficiency for the computation of dense physical systems such as the HPL benchmark.

Similar performance vs. energy efficiency analysis and projections for sparse systems based on the HPCG results look much more pessimistic. Here the two orders of magnitude lower performance delivered for sparse systems by the current supercomputing architectures strongly impact the energy efficiency.



	Nov-19	Jun-20	Nov-20	Jun-21	Nov-21
HPL-AI	445	1420	2000	2000	2000
HPL	148	416	442	442	442

Figure AB-13 HPL-AI (mixed precision) vs. HPL (double precision) Top 1 Performance

Figure AB-13 shows the performance of the HPL-AI benchmark from the year it started. In November 2019 there was only a single entry from the Summit system, and in June 2020 there were two entries from Summit and Fugaku –the top two largest systems at the time. In November 2020 there were 5 entries, in June 2021 there were 11 entries, and the latest version in November 2021 had 18 entries. Due to the small number of participations in the HPL-AI benchmark we decided to show only the Top 1 score, since the Top 10 average would not show a meaningful trend with such small number of entries since 2019. The HPL Rmax score is plotted along with the HPL-AI score, which shows the drastic difference between the performance of floating-point arithmetic when low-precision hardware can be utilized effectively. Note that the final accuracy of the solution to the linear system is identical between HPL-AI and HPL, since HPL-AI uses iterative refinement to achieve double-precision accuracy.

5.3.2. Technology Needs, System and Architectures Impact

The technological challenges that could help drive further developments in the field of physical system simulation include:

a. Reduced Data Movement.

For several decades, significantly reducing the data movement has been one of the most important challenges towards achieving higher computer performance. Achieving higher bandwidth and lower latency for accessing and moving data—both locally (memory systems) and remotely (interconnection networks)—are key challenges towards building supercomputers at ExaFLOP/s level and beyond. Breakthrough architecture solutions addressing those challenges could potentially enable up to two orders of magnitude higher performance particularly for sparse physical system simulations. More specifically, forthcoming designs of High Bandwidth Memory (HBM) such as HBM3+ and HBM4 expected to be released between 2022 and 2024, are likely to change substantially the application performance landscape for future supercomputers.

b. Efficient Floating-Point Arithmetic.

After several decades, the IEEE 754 Standard for Floating-Point Arithmetic was renewed again in July 2019[17]. However, the level of interest in this standard has been declining following critical comments about various important aspects of IEEE 754 including wasted cycles, energy inefficiencies, and accuracy. Unfortunately, the path forward is unclear at present and may involve keeping this standard as an option at least for backward compatibility while developing and implementing novel and more efficient solutions. Several efforts to address these problems follow two main approaches:

Analysis of specific algorithms and re-writing of existing codes in order to improve the performance by using lower floating-point precision without compromising accuracy. This approach has been shown to work well but only for specific algorithms/codes, and with significant dedicated efforts for each case, such as dense matrix[18], low-rank matrix[19], and sparse matrix applications[20].

More radical approaches proposing new solutions have been under development including the Posit Arithmetic proposal[21]. This work introduces a new data type — posit — as a replacement for the traditional floating-point data type because of its advantages. For example, posits guarantee higher accuracy and bitwise identical results across different systems which have been recognized as the main weaknesses of the IEEE 754 Standard. In addition, they enable more economical design with high efficiency which lowers the cost and the consumed power while providing higher bandwidth and lower latency for memory access.

c. Low Consumed Power.

During the last two decades, further developments of computer architecture and microprocessor hardware have been hitting the so-called "Energy Wall" because of their excessive demands for more energy. Subsequently, we have been ushering in a new era with electric power and temperature as the primary concerns for scalable computing. This is a very difficult and complex problem which requires revolutionary disruptive methods with a stronger integration among hardware features, system software and applications. Equally important are the capabilities for fine-grained spatial and temporal instrumentation, measurement, and dynamic optimization, in order to facilitate energy-efficient computing across all layers of current and future computer systems. Moreover, the interplay between power, temperature and performance adds another layer of complexity to this already difficult group of challenges.

Existing approaches for energy efficient computing rely heavily on power efficient hardware in isolation which is far from acceptable for the emerging challenges. Furthermore, hardware techniques, like dynamic voltage and frequency scaling, are often limited by their granularity (very coarse power management) or by their scope (a very limited system view). More specifically, recent developments of multi-core processors recognize energy monitoring and tuning as one of the main challenges towards achieving higher performance, given the growing power and temperature constraints. To address these challenges, one needs both suitable energy abstraction and corresponding instrumentation which are amongst the core topics of ongoing research and development work. Another approach is the use of application-specific accelerators to improve the application performance, while reducing the total consumed power which in turn minimizes the overall thermal energy dissipation.

The "Physical System Simulation" application area urgently needs novel and innovative architectures that provide solutions resolving the 3rd Locality Wall challenge. This includes both novel memory systems and interconnection networks offering much higher bandwidth and lower latency. Energy efficiency indicators also need urgent improvements by at least an order of magnitude. This requirement is equally valid for both homogeneous and heterogeneous architectures (including accelerators and FPGAs) that need further comparisons and analysis. Since the application area of physical system simulations is based predominantly on floating-point arithmetic, novel architecture proposals that address floating-point processing challenges can also be expected to have substantial impact, particularly for dense system computations.

References for Section 5.3.

[1] V. Getov, e-Science: The Added Value for Modern Discovery, Computer 41(8), 30–31, IEEE Computer Society, 2008.

[2] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, S.K. Weeratunga. The NAS Parallel Benchmarks, Int. J. of Supercomputer Applications 5(3), 63–73, 1991, http://www.davidhbailey.com/dhbpapers/benijsa.pdf.

[3] C. Addison, V. Getov, A. Hey, R. Hockney, I. Wolton. The GENESIS Distributed-Memory Benchmarks. In: J. Dongarra and W. Gentzsch (Eds.) Computer Benchmarks, Advances in Parallel Computing 8, 257–271, Elsevier Science Publishers, 1993.

[4] D.H. Bailey, M. Berry, J. Dongarra, V. Getov, T. Haupt, T. Hey, R.W. Hockney, D. Walker, "PARKBENCH Report-1: Public International Benchmarks for Parallel Computers", TR UT-CS-93-213, Scientific Programming 3(2), 101–146, 1994, http://www.davidhbailey.com/dhbpapers/parkbench.pdf.

[5] V. Aslot, M. Domeika, R. Eigenmann, G. Gaertner, W.B. Jones, B. Parady, SPEComp: A New Benchmark Suite for Measuring Parallel Computer Performance, Proc. Int. WOMPAT 2001, LNCS 2104, 1–10, Springer, 2001.

[6] K. Asanovic, R. Bodik, B.C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S. W. Williams, K. A. Yelick, The Landscape of Parallel Computing Research: A View from Berkeley, TR UCB/EECS-2006-183, University of California, Berkeley, Dec. 2006, http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf .

[7] https://www.top500.org/

[8] P.M. Kogge, Data Intensive Computing, the 3rd Wall, and the Need for Innovation in Architecture, Argonne Training Program on Extreme-Scale Computing, August 2017, video: https://youtu.be/ut9sBnwF6Kw,

[9] http://www.hpcg-benchmark.org/

[10] V. Marjanović, J. Gracia, and C. W. Glass, Performance modeling of the HPCG benchmark, Proc. Int. Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, pp. 172–192, Springer, 2014.

[11] E. Carson, N.J. Higham, "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions", SIAM J. Sci. Comput., 40 (2), A817–A847, 2018.

[12] J. Dongarra, P. Luszczek, A. Petitet, "The LINPACK Benchmark: Past Present and Future", Concurrency and Computation: Practice and Experience 15(9), 803–820, 2003.

[13] E. Strohmaier, H.W. Meuer, J. Dongarra, H.D. Simon, "The TOP500 List and Progress in High-Performance Computing", Computer 48(11), 42–49, IEEE Computer Society, 2015.

[14] J. Dongarra, M.A. Heroux, P. Luszczek "High-Performance Conjugate-Gradient Benchmark: a New Metric for Ranking High Performance Computing Systems", Int. J. of High Performance Computing Applications 30(1), 3–10, 2016.

[15] https://www.hpl-ai.org/

[16] V. Getov, P.M. Kogge, T.M. Conte, "Application Performance of Physical System Simulations", In: I. Foster et al. (Eds.), Parallel Computing: Technology Trends, Advances in Parallel Computing, vol. 36, pp. 251-260, IOS Press, 2020

[17] IEEE 754-2019 – IEEE Standard for Floating-Point Arithmetic, IEEE xPlore, July 2019.

[18] A. Haidar, P. Wu, S. Tomov, J. Dongarra, Investigating Half Precision Arithmetic to Accelerate Dense Linear System Solvers, Proc.ScalA Workshop at SC'17, pp. 1–8, ACM, 2017.

[19] R. Ooi, T. Iwashita, T. Fukaya, A. Ida, R. Yokota, "Effect of Mixed Precision Computing on H-matrix Vector Multiplication in BEM Analysis", ACM-IPSJ HPC Asia 2020, pp. 92-101, 2020.

[20] A. Ahmad, et al., "A survey of numerical methods utilizing mixed precision arithmetic." arXiv preprint arXiv:2007.06674 (2020).

[21] J.L. Gustafson, I.T. Yonemoto, "Beating Floating Point at its Own Game: Posit Arithmetic", Supercomputing Frontiers and Innovations 4(2), 71–86, 2017.

5.4. CRYPTOGRAPHY

This section discusses the cryptographic primitives and cryptosystems used in embedded Internet-of-Things, or IoT, devices. IoT device security is a broad domain, and this chapter will omit some topics for the sake of brevity, focusing only on the aspects that relate to performance and energy-efficiency relevant to a roughly five-year horizon.

The benchmark used to evaluate this area is EEMBC's SecureMark™-TLS[1].

Performance Trends and Predictions

IoT device security is a balance of available resources and posture. The former can be characterized as a design competition between performance and energy delivery, since IoT devices are assumed to be energy-constrained due to either battery-only operation, or powered by limited environmental generation strategies. The latter, *security posture*, sets the functional requirements of the security solution, and has changed dramatically in the IoT space in the wake of increasing embedded threats, such as the Mirai[2] botnet which targeted hundreds of thousands of consumer gadgets.

The push of new security functionality into the IoT space inevitably devolves into the traditional "power versus performance tradeoff": once a new functionality becomes a dominant requirement, resources are then invested in optimization. In this section performance refers to the speed of the computation, and energy-efficiency refers to the Joules used to perform the computation.

In the past year, new benchmark data has finally emerged to characterize some of these optimizations, including:

- Software improvements through clever assembly language programming
- Instruction-set extensions
- Hard-IP acceleration
- Peripheral accelerators

Are ISA extensions classified as *hardware acceleration*? At EEMBC we've decided the answer is no: only hard-IP that is not accessible directly through the ISA, or exists as peripheral hardware, is considered hardware acceleration.

Performance acceleration can also improve energy-efficiency: speeding up the operation by reducing instructions or memory accesses uses less energy for the computation, but also reduces the time spent in high-power MCU modes. Dedicated hardware can improve both sides of the equation.

To measure performance and energy efficiency, EEMBC developed the SecureMark™-TLS benchmark. The research group consisted of EEMBC members: Synopsys, Arm, STMicroelectronics, Texas Instruments, Renesas, and Flex. The benchmark quantifies the energy costs for performing a particular complex, yet common, operation for secure communications known as the TLS handshake[3]. This operation commonly occurs in a web browser to authenticate communication with a remote host, and share a key for encrypted transmissions. (Note that the TLS transport protocol is not limited to TCP/IP.) When modeling this for the embedded space, the collection cryptoprimitives used in the handshake—the *ciphersuite*—reflect the most likely options for energy-constrained devices. To increase portability to a wide range of hardware, the benchmark facilitates various hardware and software optimizations through an API abstraction layer. The benchmark produces two scores: an energy score which is inversely proportional to the number of *Joules* used to complete the computation (higher is better), and a performance score, which similarly is inversely proportional to the number of *seconds* taken to complete the operation (again, higher is better). The cryptoprimitives used in SecureMark-TLS are:

- AES128-ECB
- ASE128-CCM
- SHA256
- ECDH and ECDSA on the NIST p256r1 curve

All are invoked with various input dataset sizes to reflect the real-world TLS payload sizes.

Figure AB-14, below, plots all of the data available from the SecureMark-TLS score page. Although many vendors use the benchmark, not many submit scores and prefer to keep the results internal (or under NDA). Several of the devices measured have existed for over half a decade, so it doesn't yet make sense to look at the raw data from a time-based perspective. Instead, we will focus on a tradeoff view, plotting performance against energy-efficiency.

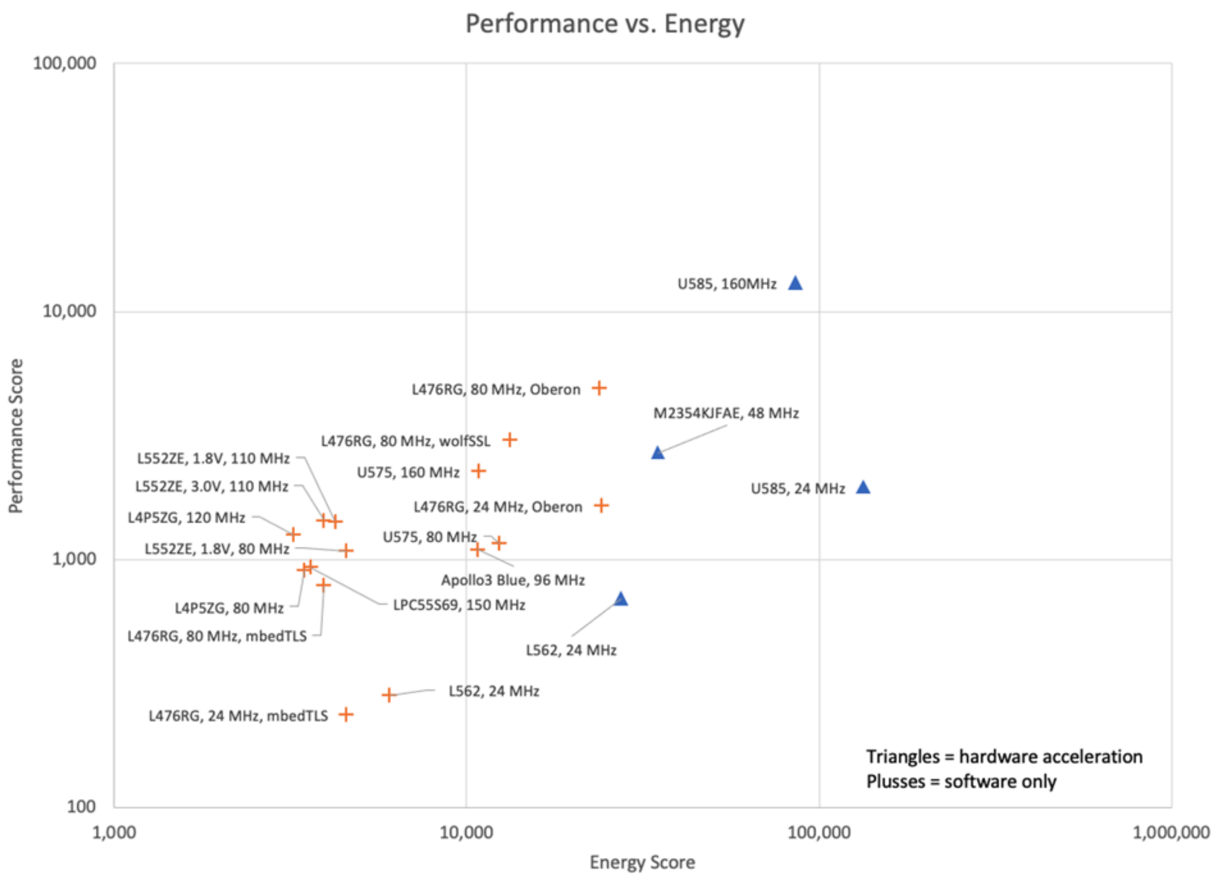


Figure AB-14 EEMBC's SecureMark-TLS benchmark results. If a device appears more than once, additional fields have been added to explain the differences. See Table AB-6 for more details on the devices (just the shorthand name for the device is used as a label to reduce visual clutter).

In this chart, hardware accelerated devices have the highest energy efficiency. All of the accelerated solutions except the STM32L562 offer acceleration of each cryptoprimitives used in the benchmark; the L562 only offers PKA acceleration for ECC, and performs AES and SHA via the ISA.

Without acceleration, scaling behaves as we'd expect with any MCU compute workload. However, it is clear that any moving to any optimized solution is going to be better than voltage or frequency scaling. For example, the L562 sees a 3.5x and 1.5x improvement in both energy efficiency and performance from moving to acceleration.

Table AB-6 A device with and without dedicated acceleration.

L562 Acceleration, 1.8 V, 24 MHz		
Mode	**Energy**	**Perf**
Without	6050	284
With	27400	699
	353%	146%

Frequency scaling has a different impact on energy efficiency when comparing the architecture of the U585 accelerator versus the Cortex-M4 in the L476. In the RISC architecture, we see a 16% improvement in energy efficiency for a 70% reduction in performance, roughly 4-to-1 performance for energy; whereas the U585 engine is roughly 1.5-to-1 performance for energy for a dedicated hardware solution.

Table AB-7 Dedicated hardware frequency scaling versus core architecture.

U585 w/Acceleration				L476RG Frequency Scaling		
MHz	**Energy**	**Perf**		**MHz**	**Energy**	**Perf.**
160	85700	13100		80	3930	787
24	134000	1950		24	4570	237
-85%	56%	-85%		-70%	16%	-70%

If we slice through Figure AB-14 for one architecture, the STMicroelectronics Cortex M4-based L476RG, we see examples of the impact of different software library optimizations as shown in Figure AB-15, below. The L476RG has no dedicated instructions or peripherals for cryptography (excluding the TRNG) so its efficiency here is entirely dictated by software optimization.

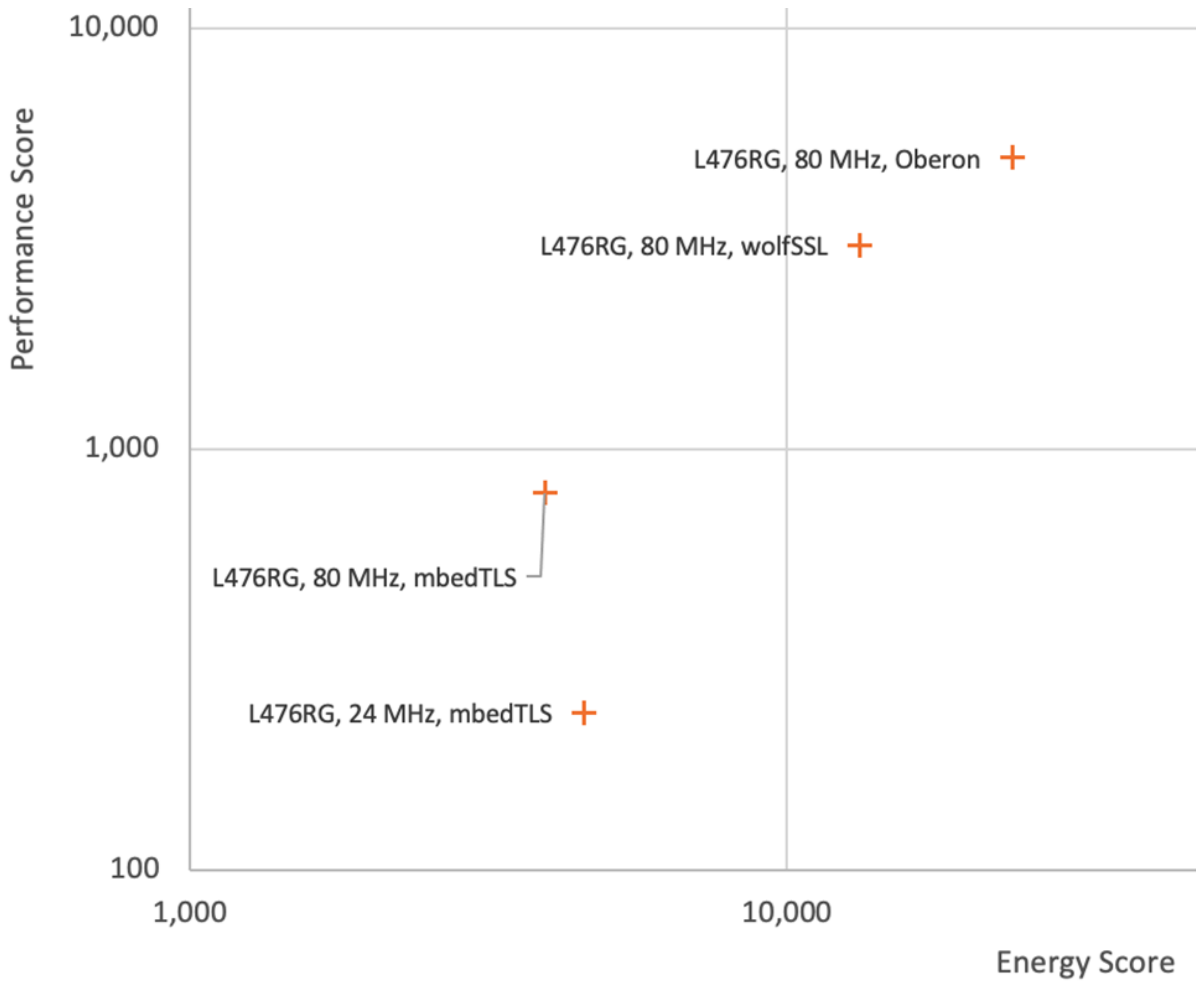


Figure AB-15 Zooming into just a single architecture with various software libraries.

At 80 MHz and 3.0 Volts, the datapoints show the relationship between three libraries: Arm's *mbedTLS*[4], Wolf's *wolfSSL*[5], and Oberon's *ocrypto*[6]. The mbedTLS and WolfSSL codecs are visible on GitHub, and looking into them reveals that mbedTLS does not offer any M4 acceleration, and wolfSSL has some lower-level M4 hand-tuned assembly optimizations. Oberon's solution is not open-source, but reportedly does additional assembly tuning by almost completely eliminating stack and heap requirements. A key observation is that both performance and energy-efficiency are improving with these optimizations. There's no reason to think that this focus on software optimization will not continue, especially as more cryptoprimitives are introduced into the mainstream.

In terms of performance predictions, we don't expect to see a repeat of the huge gains observed in this first generation of moving from software-only to hardware acceleration, but we do expect greater proliferation of dedicated hardware to all semiconductor vendors, and continued refinement on the software side. Meanwhile, new cryptoprimitives are moving into common use such as ChaCha20/Poly1305 to serve as an authenticated replacement for AES, and the twisted Edwards curve Ed25519 to replace the NIST p256r1 curve for ECC. However, cryptographic engines provide more than just acceleration, they also provide tools for secure booting, updating and key storage, and the platform itself continues to deploy new countermeasures (see the following sections).

The biggest change will likely be integration of new features at higher levels of abstraction. If one examines the software, the cryptoprimitive API is largely identical between implementations: the parameters to AES-GCM are the same regardless of implementation. However, we are seeing layers above this evolve with most vendors providing another higher-level layer for developing applications. Each vendor offering their own solution complicates the space. An attempt to standardize how these primitives interact developed by Arm called the Platform Security Architecture (PSA). We expect this evolution of the system architecture to converge on a more common vocabulary, as discussed later in this section.

5.4.1. TECHNOLOGY NEEDS

As with any data-driven research, the most important technological need is for benchmark scores to be published by vendors. Without data it is hard to glean insight and make predictions.

Beyond that, cryptographic primitives need to be fast, energy efficient, and resilient to attack from yet-invented technologies, e.g., *post-quantum cryptography*, meaning primitives that are impervious to quantum computer algorithms. NIST has been working on this through their standardization process[7].

Optimization of symmetric ciphers and message digests are common: there are plenty of ISA-optimized algorithms available, with and without dedicated extensions, and with different performance and energy-efficiency. Where things become slightly more novel is in the *public key infrastructure* (PKI) algorithms, such as Diffie-Helman key exchange, or secret generation with RSA or Elliptic Curve Cryptography. These algorithms require *big integer* math: math that uses arbitrary-sized integers in exponentiation and modulus math over integer fields. Only in the past five years or so have MCUs begun including acceleration for these operations, and we expect to see improved techniques for these operations make their way into dedicated hardware.

This can all be accomplished with current process technologies.

5.4.2. SYSTEMS AND ARCHITECTURES IMPACT

While cryptographic offloading has existed for well over a decade in the embedded space, the features have evolved from simple ISA extensions to sophisticated cryptosystems for securing keys, detecting intrusions, secure booting and provisioning, and a variety of other countermeasures. The options available to ODMs and integrators covers quite a large space, with a variety of options and caveats. This begs the question as to whether this proliferation of different techniques and approaches are obfuscating an already complex area, or simply exploring the solution-space thoroughly.

The cryptosystems are the source of failures we read about most commonly. By comparison, primitives are rarely found to be flawed or compromised, but it does happen (SHA1[8], MD5[9]). A mistake in a protocol, a feature that leaves a surface vulnerable, these are much harder to identify upfront, and usually are only discovered via bug-bounties, white-hat researchers, or is more likely the case: black-hat exploits. Part of this is due to the complexity of implementing a cryptosystem.

Here are some examples of various high-level cryptographic packages offered by several silicon vendors to facilitate in easing the complexity of implementing security protocols:

- Arm TrustZone, which covers v8 ISA instructions, CryptoCell, PSA and mbedTLS
- Renesas SCE9 Secure crypto Engine[10]
- Synopsys ARC CryptoPac ISA instructions and Enhanced Security Package[11]
- STMicroelectronics Cryptographic Service Engine peripheral
- Silicon Labs CRYPTO, CRYPTOACC, and Secure Engine Peripherals for mbedTLS[12]

Power-constrained IoT devices live in the wild, and rarely are connected to the internet directly. Typically, they interface to a *gateway* through a very low-power wireless radio such as Bluetooth, ZigBee, or low-power 802.11. A gateway is a kind of smart router that connects to the internet and manages pools of local IoT devices. Securing those devices out in the field is critical: someone can obtain the physical device, reverse engineer it, and replace it with a compromised device infiltrating the entire network much easier than, say, a laptop. Deploying these devices into use—giving them unique identities and configuring them—is called *provisioning*. This needs to be secured. Over time, the device might need a flash update. This also needs to be secured. Every time the device boots, it must come up in a secure mode. A third thing that requires on-device security.

One such feature derived from these higher-level capabilities is *secure boot*, it has a variety of implementation strategies depending in the security posture (think perf/energy tradeoffs again). The simplest boot scenario is to simply boot without any security checks at all. Clearly this not feasible in just about any scenario today. Here are three common strategies for a secure boot implementation:

1. A simple hash. On each boot the device checks the hash of the firmware to make sure the image is not corrupted. Obviously if the hash itself is compromised this fails, and if the hash is stored in secure memory, the image can never be updated without updating the secure memory too. This simple solution only requires a hash function, like SHA256.

2. Signed image. A signed firmware differs from a hash in that a public key performs authentication. This key is stored on the device secure memory to verify the firmware is valid, although the content of the firmware is visible to a hacker. During bootup the image signature is checked, and requires typically an RSA or ECDSA function.
3. Signed and encrypted image, factory key schedule. In this scenario the firmware image is encrypted and signed, and the key resides in secure device memory. This requires an RSA or ECDSA function, and also a block cipher. If executing from flash, the blocks need to be decrypted on the fly, or if there is enough RAM, the entire image can be decrypted into RAM and executed from there.

This is just a small set of examples. There are many other strategies for secure boot that can be constructed from a set of cryptoprimitives and there is no one agreed-upon solution. The point being: there is an architectural need to simplify and converge-on industry-standard practices for these additional requirements. While IETF and NIST have provided guidance, the vendor-supplied tools and techniques vary significantly. We expect the cryptosystems and their tooling to continue adding additional protocols for securing deployed devices, hopefully with consistent design architectures and methodologies. In addition, EEMBC is already working on its second-generation SecureMark benchmark to explore more deeply into applied uses of cryptosystems, rather than focusing just on primitives.

References for Section 5.4

[1] P. Torelli. "Introduction to SecureMark." EEMBC.org. https://www.eembc.org/securemark (accessed Jan. 5, 2021).
[2] "Mirai (Malware)." Wikipedia.org. https://en.wikipedia.org/wiki/Mirai_(malware) (accessed Jan. 5, 2021).
[3] T. Dierks. "The Transport Layer Security (TLS) Protocol Version 1.2." IETF.org. https://datatracker.ietf.org/doc/html/rfc5246 (accessed Jan. 5, 2021).
[4] "Arm mbedTLS repository." GitHub.com. https://github.com/ARMmbed/mbedtls (accessed Jan. 5, 2021).
[5] "WolfSSL repository." GitHub.com. https://github.com/wolfSSL/wolfssl (accessed Jan. 5, 2021).
[6] "Oberon ocrypto." Ocrypto.ch. https://www.ocrypto.ch/ (accessed Jan. 5, 2021).
[7] "PQC Standardization Process: Third Round Candidate Announcement." NIST.gov.\https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement (accessed Jan. 5, 2021).
[8] B. Schneier. "SHA-1 Broken." Schneier.com. https://www.schneier.com/blog/archives/2005/02/sha1_broken.html (accessed Jan. 5, 2021).
[9] P. Selinger. "Collisions in the MD5 cryptographic hash function." mscs.dal.ca. https://www.mscs.dal.ca/~selinger/md5collision/ (accessed Jan. 5, 2021).
[10] "Installing and Utilizing the Cryptographic User Keys using SCE9." Renesas.com. https://www.renesas.com/us/en/document/apn/installing-and-utilizing-cryptographic-user-keys-using-sce9-application-project?language=en&r=1353811 (accessed Jan. 5, 2021).
[11] "CryptoPack: Extensions for Cryptographic Software Acceleration." Synopsys.com. https://www.synopsys.com/dw/doc.php/ds/cc/arc_cryptopack-ds.pdf (accessed Jan. 5, 2021).
[12] "Cryptographic Hardware Acceleration Plugins for mbedTLS." SiLabs.com. https://docs.silabs.com/mbed-tls/latest/group-sl-crypto (accessed Jan. 5, 2021).

5.5. Emerging Area: Video Codec

5.5.1. Performance Trends and Prediction

As mobile devices such as smartphones and tablets become more ubiquitous, both supply and demand for high-resolution and high quality video content have been growing rapidly. Such video content has become a predominant bandwidth consumer, projected to take up close to 80% of the overall mobile data traffic by 2022 as shown in Figure AB-16[1]. The customer appetite and expectation for fast, ultra-high-quality video will continue to grow.

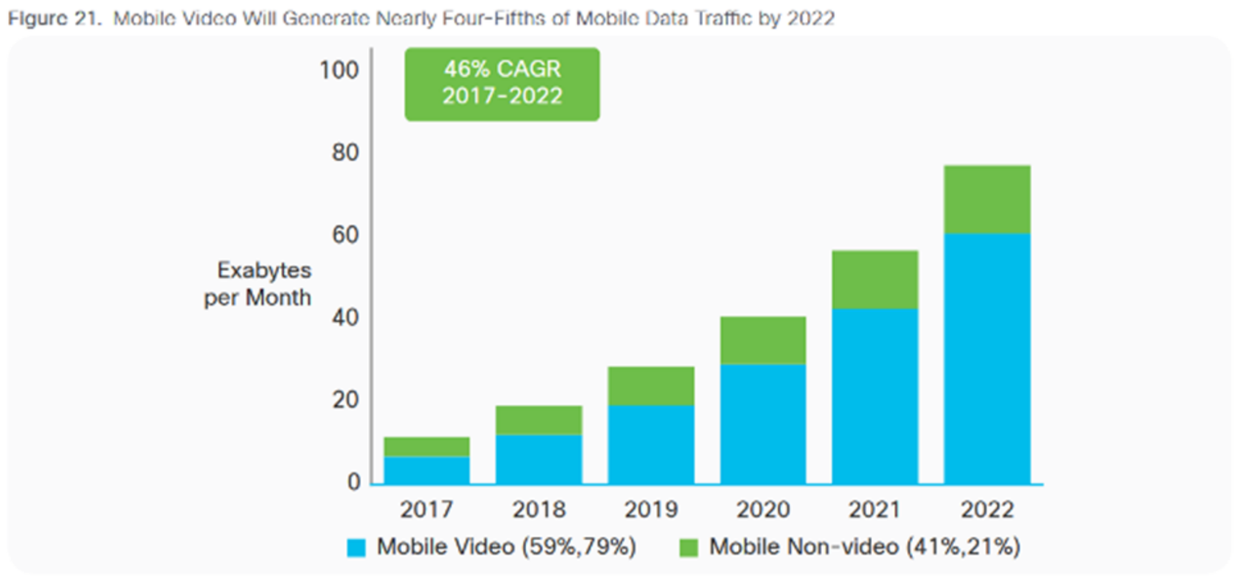


Figure AB-16 Mobile video will generate nearly four-fifths of mobile data traffic by 2022

Naturally, there is a strong demand for high efficiency video compression technology, which led to an effort to create an open video format. In 2013, Google launched VP9[2], which was competitive in coding efficiency with the state-of-the-art royalty-bearing H.265/HEVC codec. In 2015, several companies including Amazon, Google, Mozilla, Netflix, Intel, and Arm formed the Alliance for Open Media (AOM), a consortium to work jointly towards a next-generation open video coding format called AV1.

AV1 is a traditional block-based frequency transform format based on Google's VP9 codec. It incorporates additional techniques that mainly give encoders more coding options to enable better adaptation to different types of input.

Big companies are adding support for AV1.

"Netflix now streaming AV1 on Android"[3]

"Facebook video adds AV1 support"[4]

In a study by Facebook [blog], AV1 consistently showed significant bandwidth savings across various resolutions compared to x264 (main and high config) and libvpx-vp9 as shown in Figure AB-17.

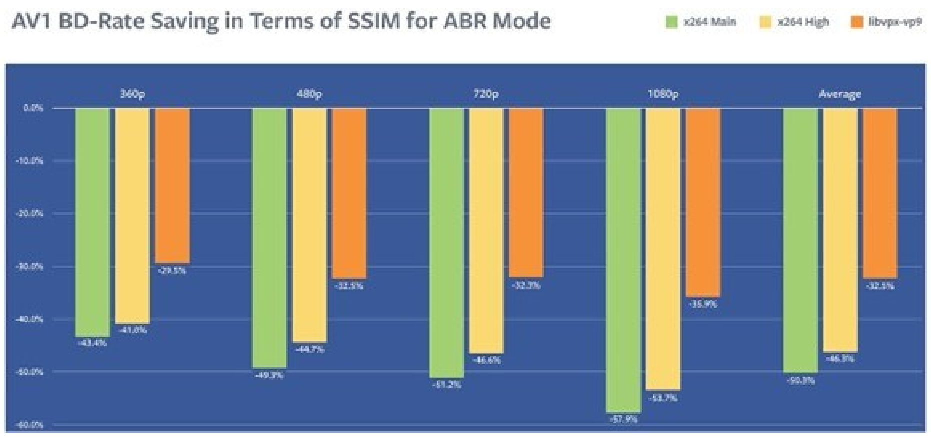


Figure AB-17 AV1 BD-rate saving in terms of SSIM for ABR mode against x264 main, x264 high and libvpx-vp9[5]

From a study presented in [ICIAR19] by the University of Waterloo, Figure AB-18 shows the RD curves of AVC, VP9, HEVC, AVS2 and AV1 encores for 540p, 1080p and 2160p resolutions for two sample videos. AV1 achieves the highest bitrate savings for high motion content among competing codecs. This is enabled by AV1's superior motion prediction schemes through warped motion, global motion tools and more reference frames.

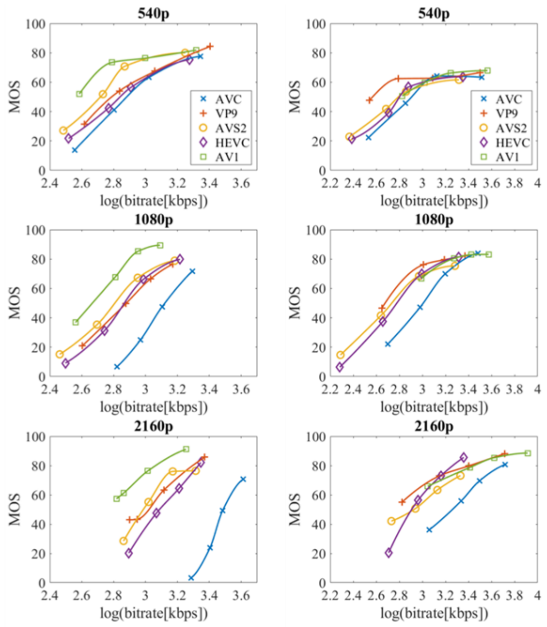


Figure AB-18 RD curves of AVC, VP9, HEVC, AVS2 and AV1 encoders for 540p, 1080p, and 2160p resolution for Tears of steel (left) and Barbecue (right)

However, the same study compares the relative encoder complexity of these codecs measured on an Intel E5-1620 CPU with 32GB memory. shown in Table AB-8. Across three resolutions, AV1 is more than two orders of magnitudes more complex compared to AVC and is more than 50 times more complex than AVS2, which is the next most complex codec.

Table AB-8 Encoder relative complexity vs. AVC at 3 resolutions

	AVC	HEVC	AV1	VP9	AVS2
2160p	1	4.2810	590.74	5.2856	9.8568
1080P	1	4.7314	546.19	6.6286	10.0401
540P	1	5.2805	806.15	5.2572	11.7716

5.5.2. TECHNOLOGY NEEDS, SYSTEM AND ARCHITECTURE IMPACT

A video is a sequence of frames which are divided into one or more tiles. Tiles are divided into large squares called superblocks. Each block may be predicted by neighboring blocks in the same tile, or by a motion projection of regions in other frames. The core of the video encoder is deciding how to partition the blocks, which prediction mode and reference to use for each block, and how to transform the difference.

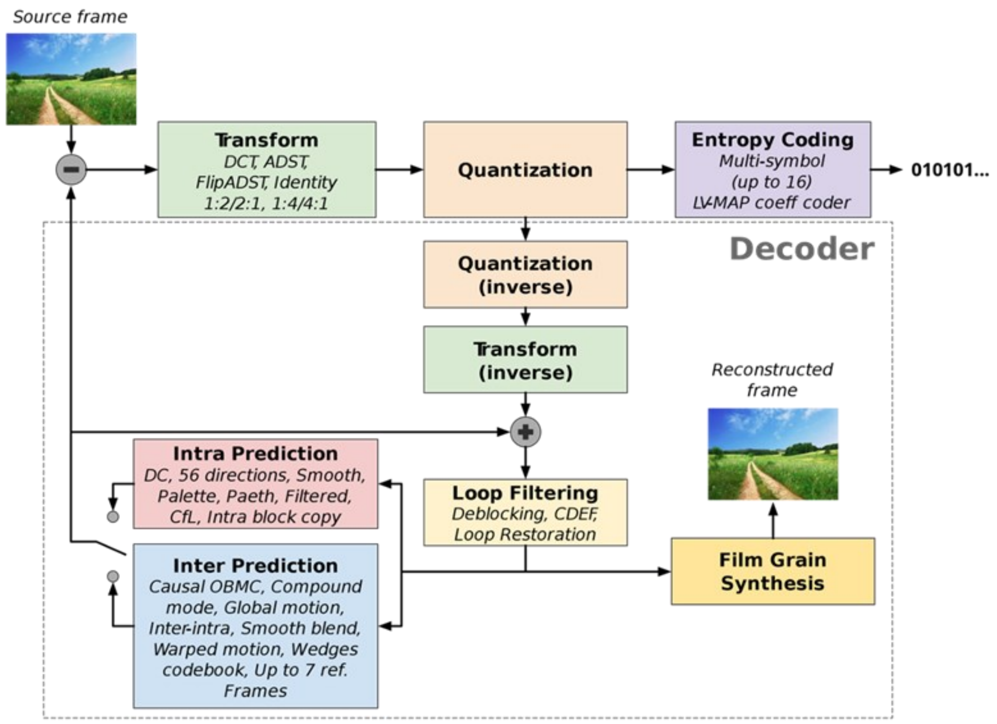

Source frame
Transform
DCT, ADST, FlipADST, Identity 1:2/2:1, 1:4/4:1
Quantization
Entropy Coding
Multi-symbol (up to 16) LV-MAP coeff coder
010101...
Decoder
Quantization (inverse)
Transform (inverse)
Reconstructed frame
Intra Prediction
DC, 56 directions, Smooth, Palette, Paeth, Filtered, CfL, Intra block copy
Loop Filtering
Deblocking, CDEF, Loop Restoration
Inter Prediction
Causal OBMC, Compound mode, Global motion, Inter-intra, Smooth blend, Warped motion, Wedges codebook, Up to 7 ref. Frames
Film Grain Synthesis

Figure AB-19 Processing stages of an AV1 encoder with relevant technologies associated with each stage.[6]

Despite the significant reduction in average bitrate at the same quality, the long encoding time remains a challenge for AV1 to be more widely adopted, especially to support real-time transcoding. While the development of AV1 stemmed from VP9, the increased overhead comes from additional techniques and complexity at every step of the encoding process.

For example, Figure AB-20 shows the partition tree in VP9 and AV1. VP9 uses a 4-way partition tree starting from the 64x64 level down to 4x4 level. However, AV1 not only expands the partition-tree to a 10-way structure, but also increases the largest size to start from 128x128.

For intra prediction, while VP9 uses 8 directional modes and 2 non-directional modes, AV1 has 56 direction models and 6 non-directional modes enabled, along with 5 recursive filtering modes and a chroma-only mode, in order to exploit more varieties of spatial redundancy in directional textures.

Similarly, there are many more added complexities in AV1 in stages such as inter prediction, transforms, quantization, entropy coding, and filtering. More details of these added complexities can be found in [7].

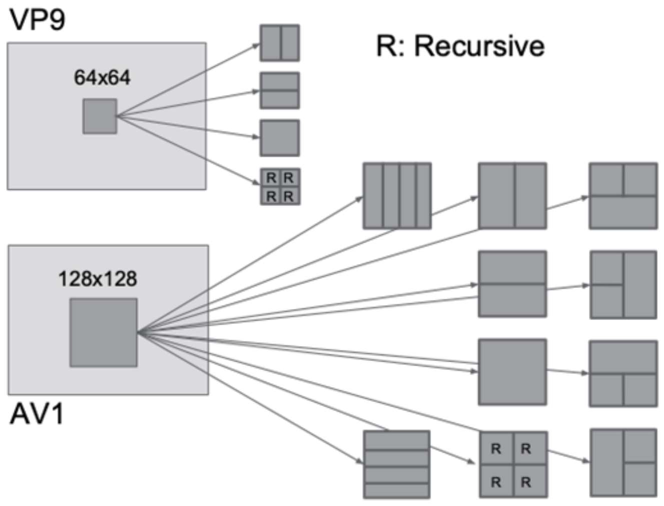


Figure AB-20 Partition tree in VP9 and AV1

From a CPU workload characterization perspective, there has not yet been a detailed study that explores the bottlenecks of these added algorithmic complexities accompany. Characterizations such as [IISWC 2020][8] can be repeated on AV1. [IISWC 2020][8] conducted a detailed analysis and microarchitectural sensitivity study while sweeping some transcode parameters in x264. Figure AB-21 shows the trade-off while turning two knobs, namely ref and crf.

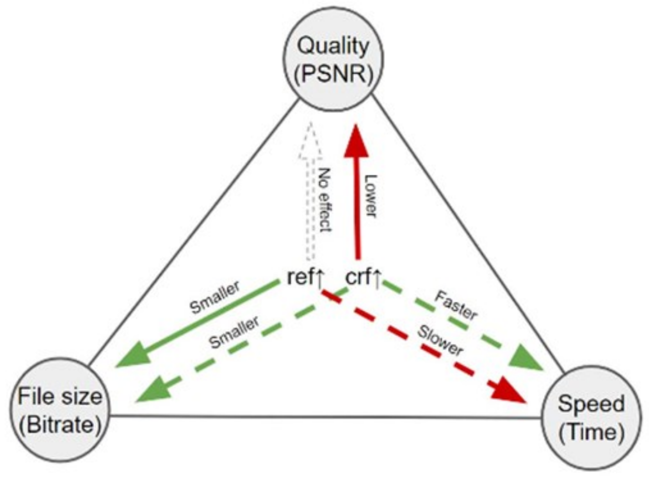


Figure AB-21 Transcoding speed, video quality, and file size triangle

Note: It shows the effects of increasing crf and refs on the three metrics. A green line denotes a positive impact, a red line represents a negative impact, a solid line demotes an active impact (purpose of changing the option), and a dotted line indicates a passive impact (side effect).

Figure AB-22 shows profiling results showing inefficiencies in different microarchitecture resources for those knobs.

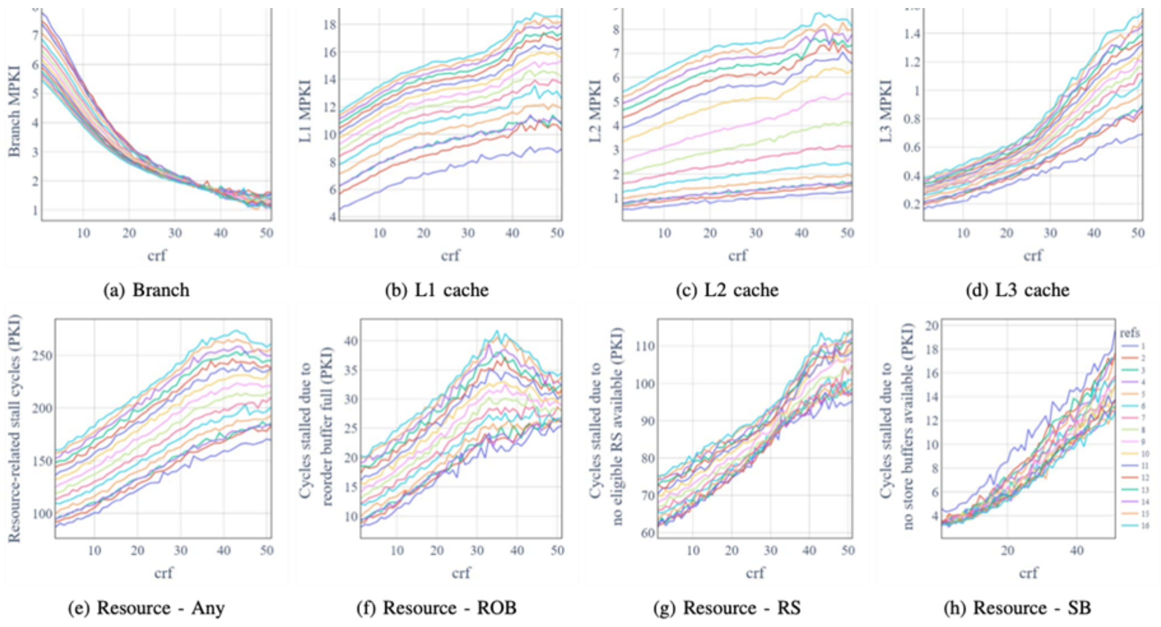


Figure AB-22 Profiling results showing inefficiency in different microarchitecture resources for different values of transcoding parameters, crf and refs.

A similar study for AV1 would help identify CPU bottlenecks and lead to potential optimizations.

[ASPLOS 2018][9] apply Processing-In-Memory (PIM) on VP9 encoding/decoding.

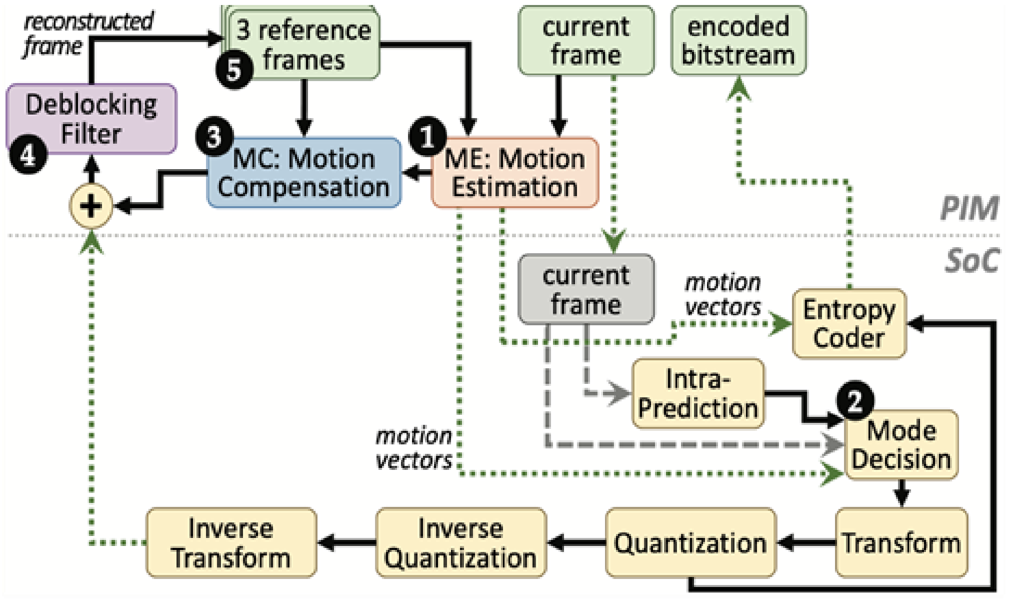


Figure AB-23 Modified VP9 encoder with in-memory ME

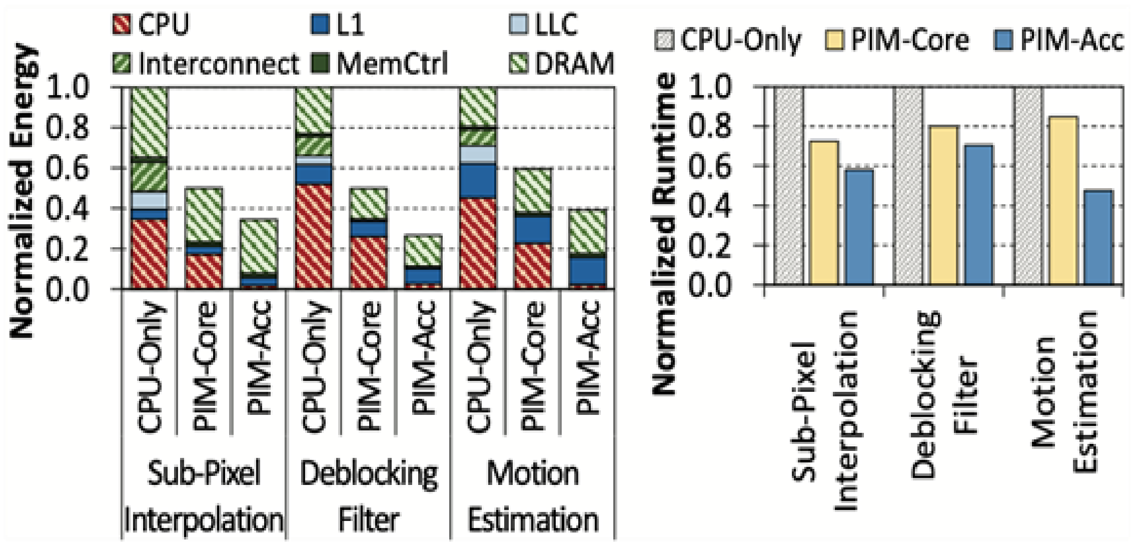


Figure AB-24 Energy (left) and runtime (right) for all video kernels, normalized to CPU-only, using 100 4K frames for decoding and 10 HD frames for encoding[10]

Moreover, dedicated hardware accelerators can be developed to significantly improve the encoding time. Google recently developed VPU [ASPLOS][9].

References for Section 5.5.

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022", Cisco, February 2019, http://media.mediapost.com/uploads/CiscoForecast.pdf

[2] Google VP9: https://developers.google.com/media/vp9

[3] "Netflix now streaming AV1 on Android", https://techcrunch.com/2020/02/06/netflix-av1-android/, Feb 2020

[4] "Facebook video adds AV1 support", Facebook blog post, https://engineering.fb.com/2018/04/24/video-engineering/facebook-video-adds-av1-support/, April 2018

[5] "AV1 beats x264 and libvpx-vp9 in practical use case", Facebook blog post, https://code.facebook.com/posts/253852078523394/av1-beats-x264-and-libvpx-vp9-in-practical-use-case/, April 2018

[6] https://en.wikipedia.org/wiki/AV1#/media/File:The_Technology_Inside_Av1.svg

[7] Chen, Yue, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker et al. "An overview of core coding tools in the AV1 video codec." In 2018 picture coding symposium (PCS), pp. 41-45. IEEE, 2018.

[8] Chen, Yuhan, Jingyuan Zhu, Tanvir Ahmed Khan, and Baris Kasikci. "CPU microarchitectural performance characterization of cloud video transcoding." In 2020 IEEE International Symposium on Workload Characterization (IISWC), pp. 72-82. IEEE, 2020.

[9] Boroumand, Amirali, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim et al. "Google workloads for consumer devices: Mitigating data movement bottlenecks." In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 316-331. 2018.

[10] Xiph.Org Foundation, "Derf Video Test Collection," https://media.xiph.org/ video/derf/.

[11] en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2017/03/22/deep-learning-inference-on-p40-gpus

5.6. MACHINE LEARNING FOR SCIENCE

Use of machine learning for science applications is a vast topic. The recent Department of Energy townhalls that are focused on this particular topic and the associated report covers this topic in substantial detail [1]. The use of machine learning in science is possible in several areas of interest to domain scientists such as chemistry and materials [2,4,6], high energy physics [7], nuclear physics, biology and life sciences [5], energy and infrastructure, climate science [3], and computer science. The use of AI/ML in many of these domains have the potential for distinct needs for devices and systems. Instead of analyzing the needs of each one of the fields individually, we look at the potential use of AI/ML in a science pipeline as represented by an abstract multiphysics pipeline shown in Figure AB-25.

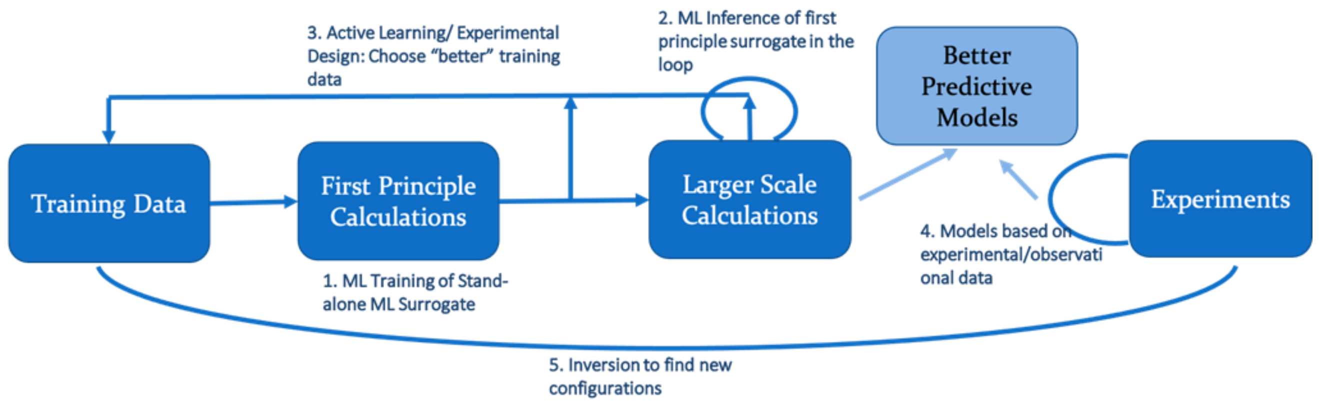

3. Active Learning/ Experimental Design: Choose "better" training data
2. ML Inference of first principle surrogate in the loop
Better Predictive Models
Training Data
First Principle Calculations
Larger Scale Calculations
Experiments
4. Models based on experimental/observational data
1. ML Training of Stand-alone ML Surrogate
5. Inversion to find new configurations

Figure AB-25 Use of ML within an Abstract Multiphysics Pipeline

This abstract model shows different possible places machine learning could be used in a modeling and simulation workflow. The key steps to focus on are the presence of multiple scales of physics as this is very common in several of the modeling and simulation workflow. Machine learning could be used to build a surrogate for first principle calculations, as part of the inference within a larger scale calculation using the ML trained surrogate, as part of an active learning or experimental design loop to generate new training data and train new models. In addition to these modeling and simulation use cases, machine learning could also be used to build models using data from experiments or physical assets that we are trying to model. Machine learning finally could also be used to connect the physical assets to digital models to generate new training data. There are a lot of similarities between these use cases. However, there are also unique requirements for each of these use cases that must be addressed in the roadmap for the devices and systems. This report highlights several examples in each of these use cases and use them to highlight the future needs in systems.

As one considers ML for Science use cases it is important to also understand that there is wide variety in what type of machine learning approaches are popular. In fact, there is a continuum of machine approaches that are used within this domain that have slightly different requirements on devices and systems. One such continuum is shown in Figure AB-26 with physical models such as finite element methods, density functional theory, in one end of the spectrum and fully data driven models in the other end of the spectrum. Several "physics informed machine learning" approaches exist in between these two extremes. The most common approach is to incorporate physics in the loss function. However, several new approaches are being developed where physics is explicitly handled in the machine learning model or it is encoded in the input. One example on how such a spectrum would impact device and system of the future is where the loss function itself requires a physics solve. Such a requirement might be better served by a system that is suitable for a traditional solve and a machine learning model or by a heterogeneous node with several specialized accelerators.

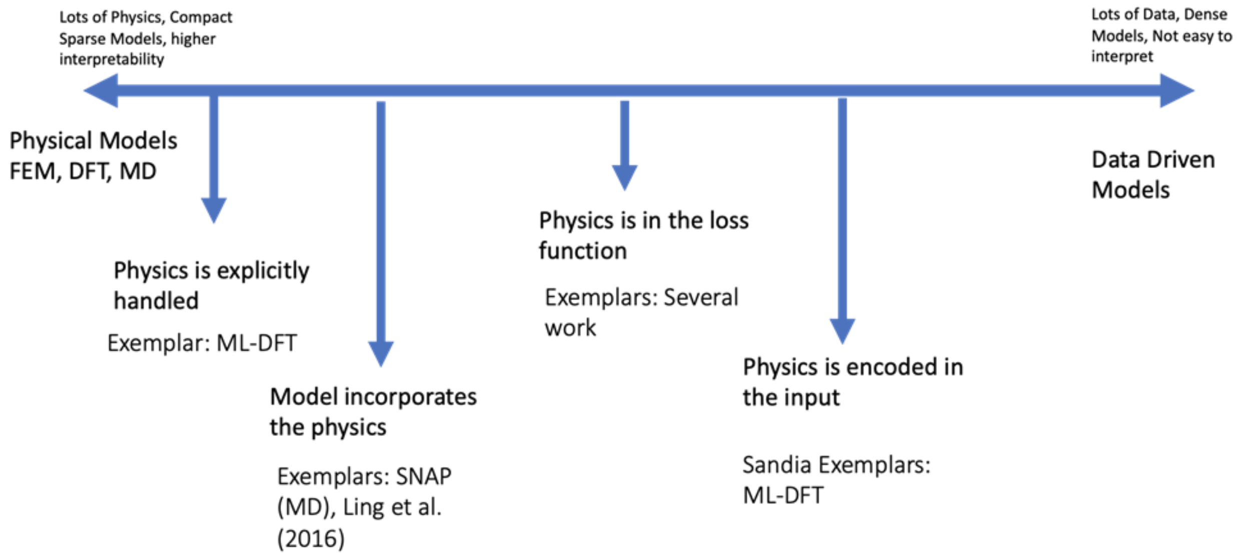

Lots of Physics, Compact Sparse Models, higher interpretability
Lots of Data, Dense Models, Not easy to interpret
Physical Models FEM, DFT, MD
Physics is explicitly handled
Exemplar: ML-DFT
Model incorporates the physics
Exemplars: SNAP (MD), Ling et al. (2016)
Physics is in the loss function
Exemplars: Several work
Physics is encoded in the input
Sandia Exemplars: ML-DFT
Data Driven Models

Figure AB-26 Spectrum of machine learning approaches on ml for science use cases

5.6.1. Large Scale Inference within Physics Simulations

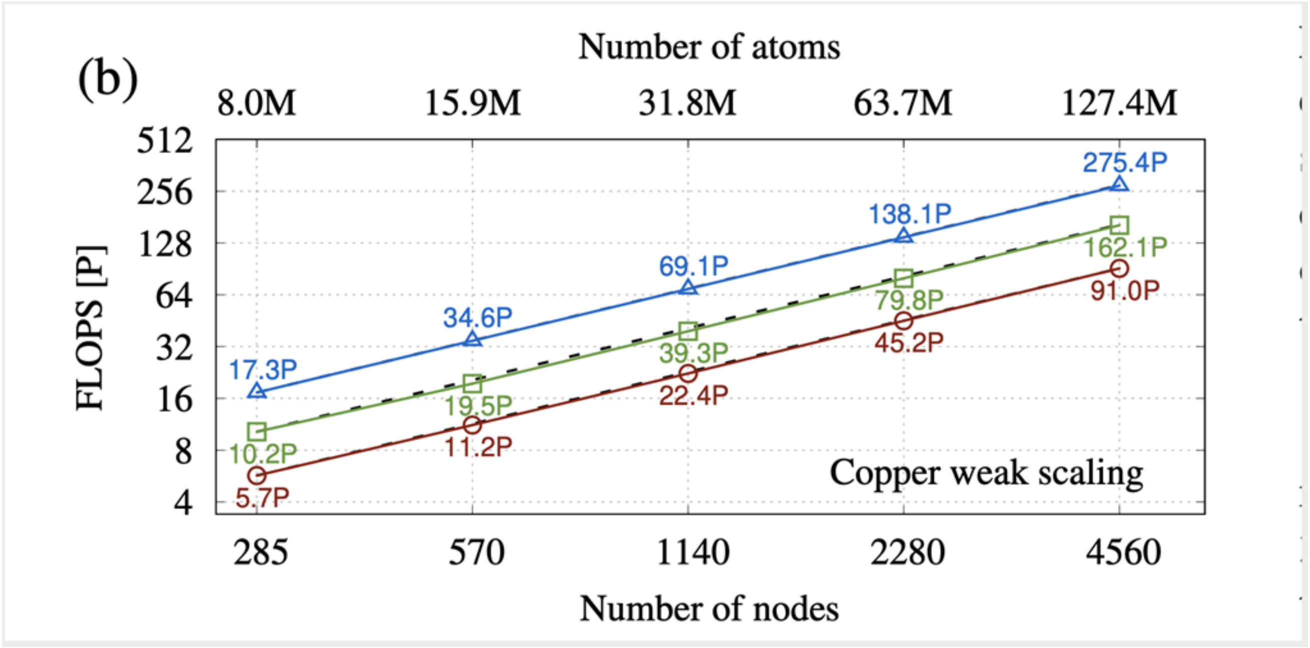


Figure AB-27 Gordon Bell Prize winning materials calculation using machine learning that reaches 275 pflops using thousands of GPUs [2]

The first example we start with is the problem of using a trained machine learning model for a given material within large scale materials calculation. Figure AB-27 shows performance of a recent Gordon Bell prize winning DeepMD code for a Copper system [2]. The key part of this simulation is using a trained ML model within a large scale MD simulation to be able to solve a science problem at a scale that was not possible before. The results show excellent scaling up to 4560 nodes with thousands of GPUs. Fast inference integrated with the molecular dynamics simulation poses unique challenges to hardware as the inference is tightly integrated with other steps within physics simulation. As the molecular dynamics simulations are tightly integrated with the inference, any system design would have to consider accelerating both the science simulation and the inference portion, efficient data accesses for both, and data sharing/transfer between the physics simulation and machine learning inference. The parallel scalability derives from the science simulation and the inference can be a local inference in most such examples when the model itself is able to fit on a single device. The growth in the size of the models have to be met by improved memory capacity, out-of-memory algorithms, or distributed inference.

5.6.2. TRAINING WITH LARGE AMOUNTS OF DATA FOR SCIENCE SIMULATIONS

Machine Learning training needs have been a key constraint on how much data can be used, how large of a model can be trained, and how much hyperparameter optimization can be done to tune the model. Figure AB-28 shows recent work from OpenAI on the amount of compute required to train several AI models. This graph shows that the amount of compute needed has doubled every 3-4 months.

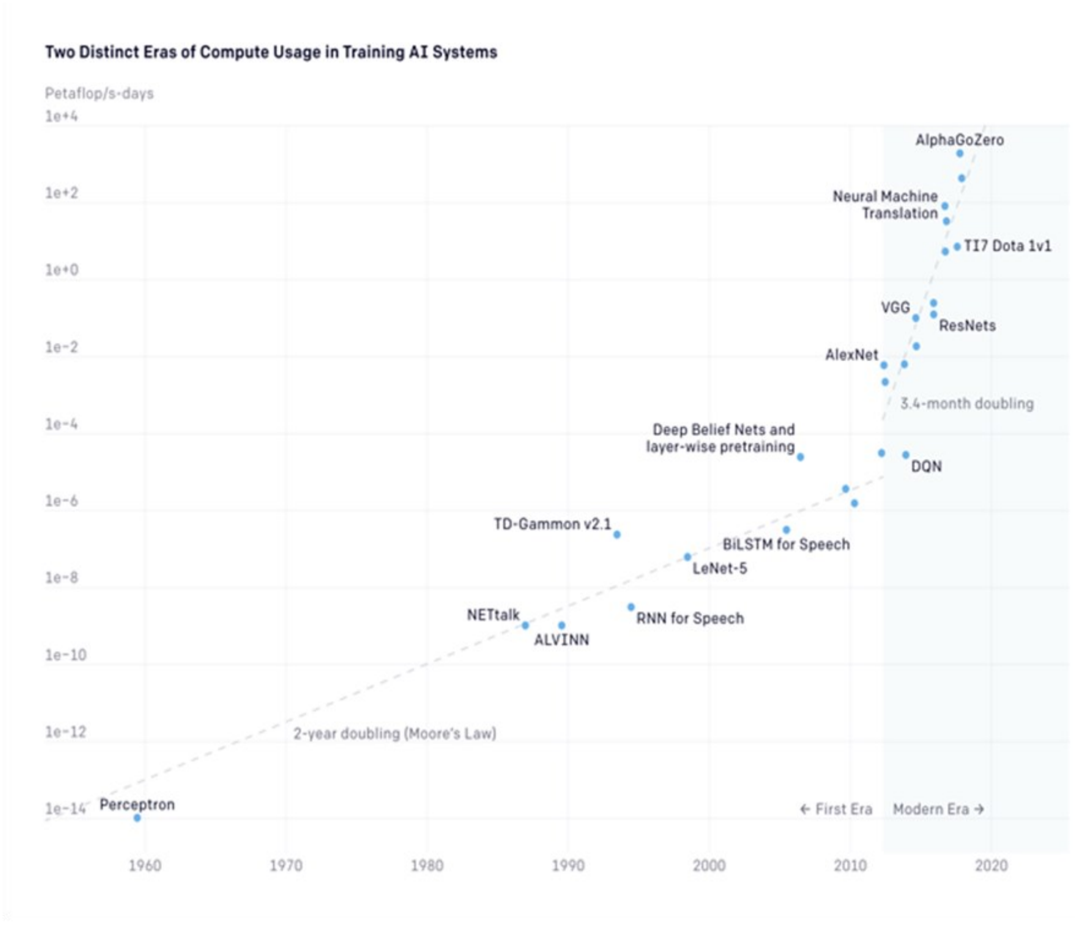


Figure courtesy: https://openai.com/blog/ai-and-compute/

Figure AB-28 The amount of compute needed for AI training

While the ML for science simulation needs for training are on different magnitude at this time, partially because of still being nascent compared to AI models used in industry, the growth trend for ML training needs are not very different based on anecdotal evidence. We show two examples from recent work that demonstrate the requirements from science applications.

Recent Gordon Bell prize winning work used 27K GPUs for training the climate simulations to achieve 1.13 EF/s of FP16 performance [3]. This work is one example of the need for large scale compute for AI training for climate simulations.

On the other side of the spectrum, there is also a need for using large amounts of data. Recent work on machine learning for density functional theory calculations [4] uses 440+ GB of data for two temperatures of one material. When expanded to multiple materials, temperatures, densities, and temperatures this data set is expected to be 100s of terabytes of training data. Such unique science use cases bring in important requirements to machine learning training hardware where a workflow that can handle terabytes of training data is important. It is also important to note that these science use cases require FP32 precision for their accuracy needs. Hardware that support FP32 will be important for ML for science use cases.

5.6.3. MACHINE LEARNING ON EXPERIMENTAL DATA

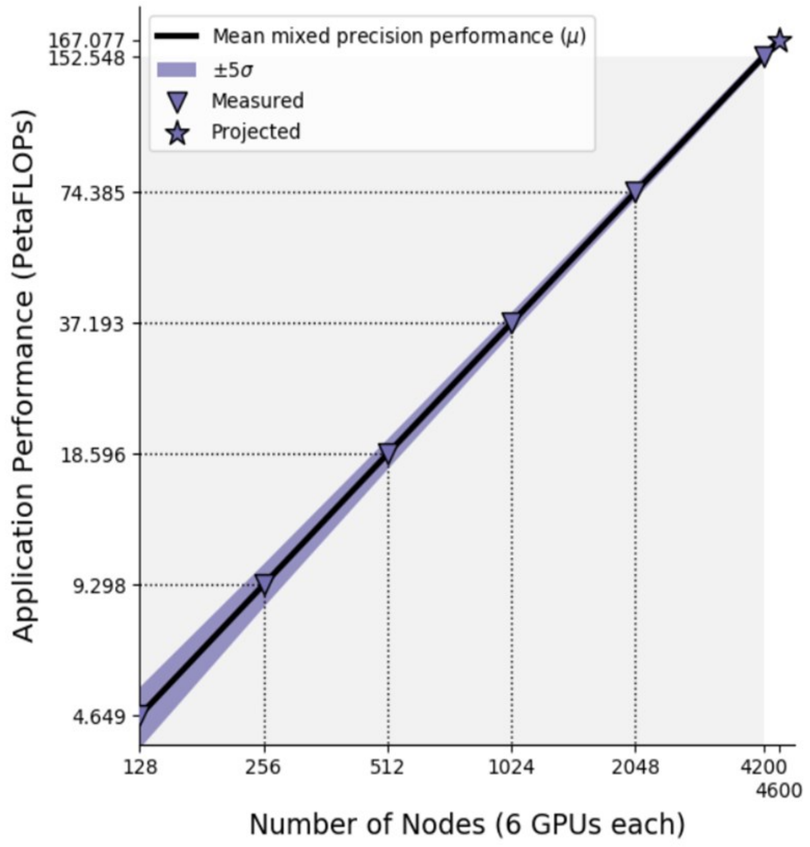


Figure AB-29 Scaling of Application performance for electron microscopy data on 25,200 GPUs

Figure AB-29 demonstrates scaling of an ML for science use case with experimental data. This was another Gordon Bell finalist few years ago [5]. This example comes from the electron microscopy application. However, number of other examples from science use cases have similar needs. The example here shows the results of not just ML training, but the hyperparameter tuning, and a genetic algorithm to choose a network. The scaling of this application to 25,200 GPUs results in 167 PFlop calculations. This application again demonstrates the need for FP32 calculations, and mixed precision calculations from the ML for science use cases.

References for Section 5.6

[1] https://www.anl.gov/ai-for-science-report
[2] Jia, Weile, Han Wang, Mohan Chen, Denghui Lu, Lin Lin, Roberto Car, E. Weinan, and Linfeng Zhang. "Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning." In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-14. IEEE, 2020.
[3] Thorsten Kurth, Sean Treichler, Joshua Romero, Mayur Mudigonda, Nathan Luehr, Everett Phillips, Ankur Mahesh et al. "Exascale deep learning for climate analytics." In SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 649-660. IEEE, 2018.
[4] J. Austin Ellis, L. Fiedler, G. A. Popoola, N. A. Modine, J. A. Stephens, Aidan P. Thompson, A. Cangi, and S. Rajamanickam. "Accelerating finite-temperature Kohn-Sham density functional theory with deep neural networks." Physical Review B 104, no. 3 (2021): 035120.
[5] Patton, Robert M., J. Travis Johnston, Steven R. Young, Catherine D. Schuman, Don D. March, Thomas E. Potok, Derek C. Rose et al. "167-pflops deep learning for electron microscopy: from learning physics to atomic manipulation." In SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 638-648. IEEE, 2018.
[6] https://doi.org/10.14278/rodare.1107
[7] Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He et al. "CosmoFlow: Using deep learning to learn the universe at scale." In SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 819-829. IEEE, 2018.

5.7. Emerging Area: Internet of Things Applications

The Internet of Things (IoT) stands for a network of large numbers of embedded devices that operate by communicating with other devices on the internet, of increasing capabilities. The term was coined by Peter T. Lewis in 1985: "The Internet of Things, or IoT, is the integration of people, processes and technology with connectable devices and sensors to enable remote monitoring, status, manipulation and evaluation of trends of such devices."[1] An early example was from 1982 when grad students at Carnegie Melon University connected a soft drink vending machine to the internet to report on contents and temperature. Today common examples include "smart home" devices such as video surveillance cameras and home temperature controls. Other application areas range from remote medical monitoring such as for elder care, through transportation management, to preventive maintenance and weather monitoring. Figure AB-30[2] lists the results of one survey of such areas from 2020.

One estimate is that the number of IoT devices today exceeds 2 billion, and will grow to 6 billion in 5 years as pictured un Figure AB-31.[3] In this figure "NB-Iot" and "Cat M1" refer to low-power wide area wireless network (LPWAN) cellular technology as defined by the standards group 3GPP [4], with speeds between 200 kbps and 400 kbps, and designed for machine-to-machine communication where connections are made dynamically when devices have data to share, or a internet host needs to establish communication. Other estimates [Tanweer, 2018][5] place the number of connected devices much higher, as much as 35 billion today and 75 billion by 2025.

Given the ubiquitous nature of IoT devices, power and energy consumption are clear priorities for their implementation, both in internal processing and in communication with external networks. There are emerging benchmarks to measure at least communication properties. One is IoTMark [6] which is aimed at energy efficiency, with two defined metrics: IoTMark-BLE for BlueTooth and an upcoming IoTMark-Wi-Fi for Wi-Fi connections. IoTMark-BLE models an I2C sensor, processor, and BlueTooth radio, from wakeup, through "advertise" (look for connectivity), to connections, and back to sleep. The reported "IoTMark" metric is a weighted composite for power during each state.

More processing centric benchmarks are also being developed. For example, [Shukla, 2017][7] identifies four performance metrics: latency, throughput, jitter, and CPU and memory utilization, along with almost 30 microkernels that together can be used with 4 streaming workflows to match up with several possible IoT scenarios.

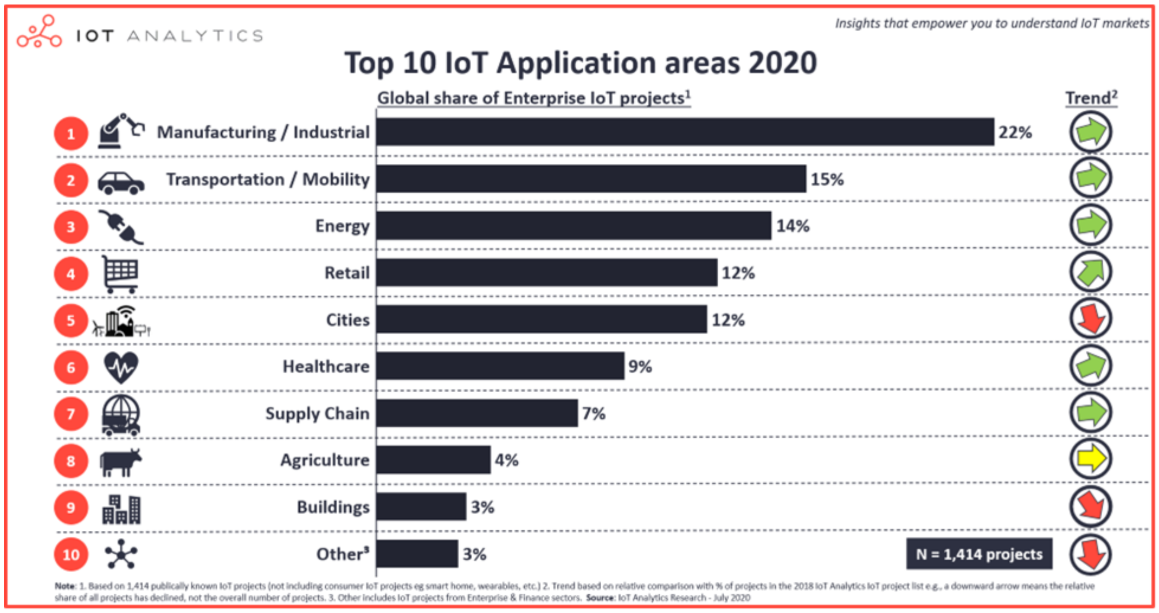
There is a growing desire to infuse IoT devices with some form of AI, for either autonomous control or data analysis. Such IoT "intelligence" can reside in three configurations:

1. Within the IoT device itself, such as for collision detection in an automobile sensor.
2. At the edge of the network but beyond each IoT device, such as in real-time traffic-driven traffic light controls.
3. In servers in the cloud, as in real-time vehicle fleet route planning.

The second of the above brings up a mode of computing half way between pure IoT and Cloud computing that is of growing importance, called "fog computing." In this model, there is a layer of "fog nodes" at the edge of the conventional internet that serve as junction points for a subset of Iot devices. Unlike a cloud server, a fog node is aware of some geographical or logical set of IoT devices with which it interacts. Interaction with higher cloud-level servers is through these fog nodes.

The importance of such models has grown to the point where NIST has developed a formal conceptual model [Iorga, 2018][8], and the IEEE has developed a standard reference architecture [IEEE, 2018][9]. There is also an "OpenFog Consortium" [OPC, 2016][10] to drive the continued development of such standards. At least one set of benchmarks [McChesney, 2019][11] have been defined that use fog nodes in a variety of ways to interact with sets of IoT devices., and provides measurements on several implementations.

Future editions of the IRDS AB Chapter will seek to track the trajectory of applications once there are sufficient results from benchmark suites discussed above. Of particular and continued interest will be the ability to reduce the energy of communication, and also the ability to embed AI and machine learning functions at very low energy in leaf IoT devices. The latter in particular may benefit from mixed analog/digital "in memory" computational structures as may be enabled by emerging alternative memory technologies.



Source: IOT Analytics

Figure AB-30 Top 10 IoT Application Areas 2020

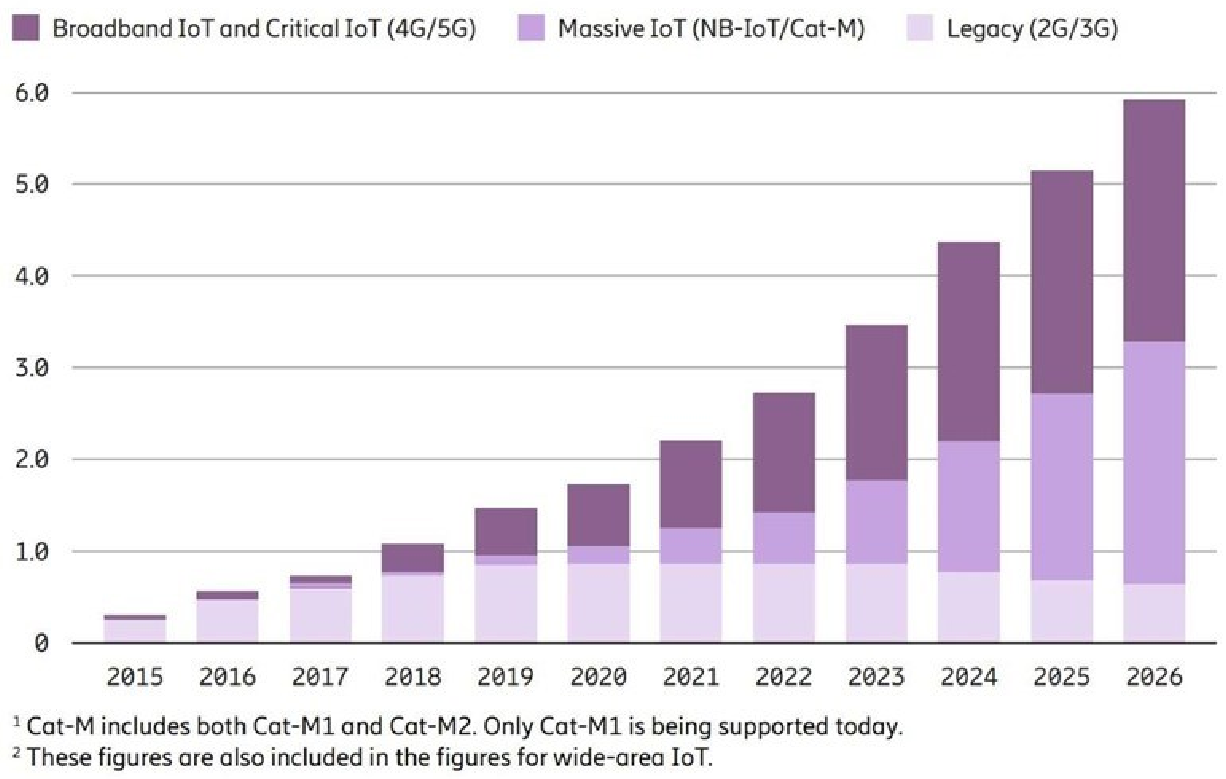


Figure AB-31 IoT Devices[12]

References for Section 5.7

[1] Speech to the Congressional Black Caucus Foundation 15th Annual Legislative Weekend in Washington, D.C. see http://www.chetansharma.com/publications/correcting-the-iot-history/

[2] https://iot-analytics.com/wp/wp-content/uploads/2020/07/Top-10-IoT-applications-in-2020-min.png

[3] https://iot.eetimes.com/wp-content/uploads/sites/2/2020/12/number-of-iot-devices.jpg

[4] https://www.3gpp.org/

[5] Alam, Tanweer. (2018). A Reliable Communication Framework and Its Use in Internet of Things (IoT). 3.

[6] https://www.eembc.org/iotmark/

[7] Shukla, A., Chaturvedi, S., & Simmhan, Y. (2017). RIoTBench: An IoT benchmark for distributed stream processing systems. *Concurrency and Computation: Practice and Experience*, *29*(21). https://doi.org/10.1002/cpe.4257

[8] Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N., & Mahmoudi, C. (2018, March). *Fog Computing Conceptual Model* . https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-325.pdf.

[9] IEEE Communications Society. (2018, June 14). *1934-2018 - IEEE standard for adoption of OpenFog reference architecture for Fog Computing*. IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8423800&tag=1.

[10] OPC Foundation. (2016, October 27). *OpenFog*. OPC Foundation. https://opcfoundation.org/markets-collaboration/openfog/.

[11] McChesney, J., Wang, N., Tanweer, A., de Lara, E., & Varghese, B. (2019). *DeFog: Fog Computing Benchmarks*. https://arxiv.org/abs/1907.10890.

[12] https://iot.eetimes.com/wp-content/uploads/sites/2/2020/12/number-of-iot-devices.jpg

6. CROSS MATRIX

The function of the Cross Matrix is to map application areas to SA-IFT-defined market driver :

- **Cloud:** This category is for server devices deployed in data centers. The term "cloud" refers to the engineering of data center scale computing operations: compute, storage, networking engineered for scale and for continuous resource redeployment and reconfiguration via APIs. Whether they are operated publicly or privately, they offer on demand, as-a-service consumption model. While they had their origins in web service; media streaming, shopping and commerce; they are increasingly broadening their applications base to big data for social networking, recommendations, and other purposes; precision medicine; training of AI systems, and high performance computation (HPC) for science and industry.

- **Personal Augmentation (PA):** Personal augmentation devices provide multiple use cases: telephony and video telephony; multimedia viewing; photography and videography; email and electronic communication; positioning and mapping, authenticated financial transactions, health and fitness monitoring, personal safety and environmental warning. Current and upcoming market drivers include: gaming and video applications; productivity applications; social networking; augmented reality and context-aware applications, and mobile commerce. Personal augmentation devices already make use of AI technologies such as personal assistants. Deployment of AI on and through personal augmentation devices will accelerate.

- **Internet-of-Things edge devices (IoT-e):** Although IoT is a broad class if computing applications spanning the server to the ultimate sensors and actuators, an IoT edge (IoTe) device is a wireless device with computation, sensing, communication, and possibly storage. The device may include one or more CPUs, memory, non-volatile storage, communication, security, and power management. It may be line powered, battery powered or utilize energy harvesting.

- **Cyber-Physical Systems (CPS):** This category encompasses computer-based control of physical devices characterized by real-time processing and used primarily in industrial control. Many cyber-physical systems are safety-critical. They interface to the systems they control via both standard and proprietary interconnects broadly know as Operational Technology (OT), where ruggedness, extended environmental capabilities, low cost have been paramount over considerations such as security and attestation.

Below in Table AB-9 is the cross matrix that was developed. (“Y” means important application area(s).)

Table AB-9 Cross Matrix of Application Area vs. Market Drivers

Application Area	Cloud	IoTe	CPS	PA
Graph Analytics	Y	Y	Y	Y
Artificial Intelligence	Y	Y	Y	Y
Physical System Simulation	Y	Y	Y	
Cryptography	Y	Y	Y	Y
Video codec	Y	Y		Y
Machine Learning for Science	Y			
Internet of Things Applications		Y	Y	Y

7. CONCLUSIONS AND RECOMMENDATIONS

Applications are the drivers of much of the nanoelectronics industry. The challenge is how to connect the trends in applications to the nanoelectronics trends and needs. This chapter of the IRDS has sought to do this by extrapolating from representative benchmark programs for each application area to performance trends over time and critical technology needs. Table AB-2 shows a summary of the findings of the AB IFT. Memory bandwidth increases are critical for all of the application areas we studied. Not all of the application areas benefit from improvement in memory latency, however. This suggests that logic in memory is not a universal solution to meeting all applications. Rather, it is most important for simulation and optimization application areas.

Another emerging trend is to use fixed-function accelerators to speed up critical applications. This improves the power efficiency of microprocessors because it obviates the need to continuously fetch and decide instructions. For example, a common application area is discrete cosine transform, or DCT. A fixed-function accelerator for DCT has the advantage over a software implementation in that it does not require the fetching and decoding of the software instructions. Two key application areas are predicted to benefit most from fixed-function acceleration: Feature Recognition, Graphics/VR/AR and Cryptographic Codec.

The process of applications benchmarking is imperfect in several senses. The AB IFT makes the following recommendations for improving the process of applications benchmarking itself. (1) The benchmarks that are being used today must continue to be used into the future. In addition, (2) investment in new benchmark development is critically important for several of the application domains studied, including Feature Recognition, Discrete Event Simulation, and Optimization. In addition to those application areas, the AB IFT sees the need for investment in benchmarks to address cryptography and security and to address multimedia/DSP workloads. Both (1) and (2) are standards activities that the IEEE Standards Association could tackle with new working groups.