INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS™

2018 UPDATE

APPLICATION BENCHMARKING

THE IRDS IS DEVISED AND INTENDED FOR TECHNOLOGY ASSESSMENT ONLY AND IS WITHOUT REGARD TO ANY COMMERCIAL CONSIDERATIONS PERTAINING TO INDIVIDUAL PRODUCTS OR EQUIPMENT.

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

The AB IFT team takes all credit (and blame) for the contents of this chapter. Thanks goes to the members of the team, who are:

Name	Representing
Geoffrey Burr	IBM
Tom Conte [chair]	Georgia Tech & IEEE RCI
Vladimir Getov	University of Westminster, UK
Peter M. Kogge	University of Notre Dame
Markus Levy	Chair of EEMBC
Satoshi Matsuoka	Tokyo Tech [SDRJ representative]
Dam Sunwoo	ARM
Josep Torrellas	University of Illinois at Urbana-Champaign

Early on, the AB team benefitted also from the input of Doug Carmean (Microsoft), Jonathan Gallmeier (AMD), Naveen Muralimanohar (HPE) and Yoshihiro.Hayashi (Renesas).

In addition, the team would also like to thank Linda Wilson for all of her help and support. We thank Paolo Gargini for his input on and help developing the concept of an AB IFT as a “front end” to the roadmap. We also thank Marilyn Wolf, chair of the SA IFT, for her many interactions facilitating the interface between the AB IFT and the SA IFT.

APPLICATION BENCHMARKING

1. 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, 10 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 are summarized below in Table AB-1.

Table AB-1 Application Areas

Application Area	*Description*
Big Data Analytics	Data mining to identify nodes in a large graph that satisfy a given feature/features
Feature Recognition	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.
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.
Physical System Simulation	Simulation of physical real-world phenomena. Typically, finite-element based. Examples include fluid flow, weather prediction, thermo-evolution. Computation is floating-point-based.
Optimization	Integer NP-hard optimization problems, often solved with near-optimal approximation techniques.
Graphics/VR/AR	Large scale, real-time photorealistic rendering driven by physical world models. Examples include interactive gaming, Augmented Reality, Virtual Reality.

DNN—deep neural network, NP—the non-polynomial complexity class of algorithms, VR—virtual reality, AR—augmented reality

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.

[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.*

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

1. **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).
2. **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. We initially planned to track a media processing application area but had a problem with (2): benchmark result availability. Although there have been significant benchmarks in this area, few have sufficient historical data to derive trends from. We initially planned to also track the performance of a cryptographic codec application area, but we found another challenge in addition to (1)– (3) and (4): much cryptographic codec work is done in fixed function ASIC accelerators. Much of this is vendor specific and, in some cases (e.g., Apple Secure Enclave), proprietary.

1.1. Current State of Technology

Application areas of interest for the IRDS are outlined in Table AB-1. These application areas were identified through dialogue between the members of both the SA IFT and AB IFT. Big data analytics is a focus of search giants. It is also vital to the business model of social network companies. Feature recognition is critical for human-computer interaction and for analysis of non-uniform, irregular data (such as used by the military and by search companies). Discrete event simulation is critical for industrial and systems engineering, computer systems engineering, telecommunications, transportation, etc. In contrast, physical system simulation is critical to product design in the automobile and aerospace industries. It is also of 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. Graphics and media processing has a large impact on consumer electronics (e.g., gaming consoles), but also telecommunications and storage industries. It is also critical for search companies.

1.2. Drivers and Technology Targets

This is a brief description and list, based on Overall Roadmap Technology Characteristics drivers for Systems (applicable systems or applications attributes) and Support Technologies (applicable device attributes).

1.3. Vision of Future Technology

All of the application areas of Table AB-1 are expected to remain important over the coming 15 years. One additional application area that is becoming similarly important is computer security. This area is difficult to track for reasons explained above in Section 1. The AB IFT continues to seek ways to capture this applications area. At this time, the AB IFT does not envision new application areas that are disjoint from or not covered by the application areas in Table AB-1. That said, suggestions from readers of this chapter for improvements are always welcome.

As outlined below in Section 3, most of the application areas can benefit from enhanced memory bandwidth. In some cases, benefits will also come from reduced memory access latency. 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.

2. 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.

3. SUMMARY AND KEY POINTS

A summary of the application area analysis is presented in Table AB-2. 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 of the microprocessors. 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, 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 the most 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**
Big Data Analytics		X	X	X	
Feature Recognition	X	X			X
Discrete Event Simulation		X	X		
Physical System Simulation	X	X	X	X	
Optimization	X	X	X		
Graphics/VR/AR	X	X			X

4. APPLICATION ANALYSIS

What follows is a detailed analysis of the five application areas presented in Table AB-1.

4.1. BIG DATA ANALYTICS

This workload class involves a variety of data mining 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].

While this domain is very popular, there are a large variety of frameworks and machine sizes. 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 such 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 (32–60 cores). 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; in V2.0 created in June 2017, however, they have 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.

Problem Sizes. The benchmark gives different problem classes, based on the approximate size of the graph. The classes are:

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

4.1.1. 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 three machines from November 2010 (which is when the Graph 500 competition started) to June 2017 (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.

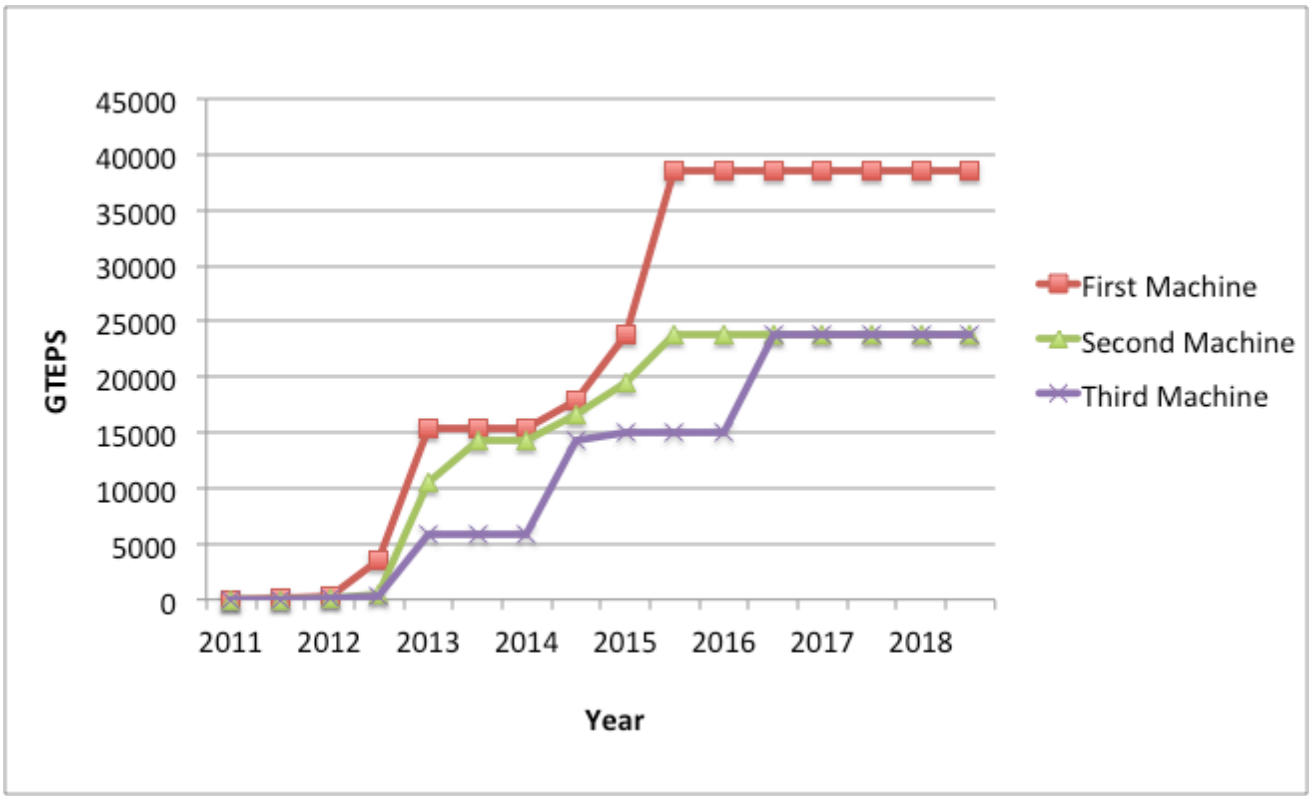


Figure AB-1 Number of GTEPS of Top Three Machines Since Inception of Graph 500 Competition

We see that the GETPS rate has been steadily increasing, although it seems to have been moving in plateaus. As of November 2018, the top three are the K Computer (Japan), the Sunway TaihuLight (China), and the DOE/NNSA/LLNL Sequoia (USA). The top three machines are invariably custom-designed machines, with custom networks and, at this point, with 1.3 to1.5 petabytes of memory.

Figure AB-2 plots the number of cores used in the top three Graph 500 machines in the same period of time. We see that two of them use around 1 to 1.5 million cores, and the other uses 10 M cores. The former use more traditional HPC-class cores, while the latter uses simpler, custom cores.

Overall, we expect that both the TEPS and the number of cores in the top three machines will continue to go up in bursts in the near future. One difficulty is that the machines are quite expensive.

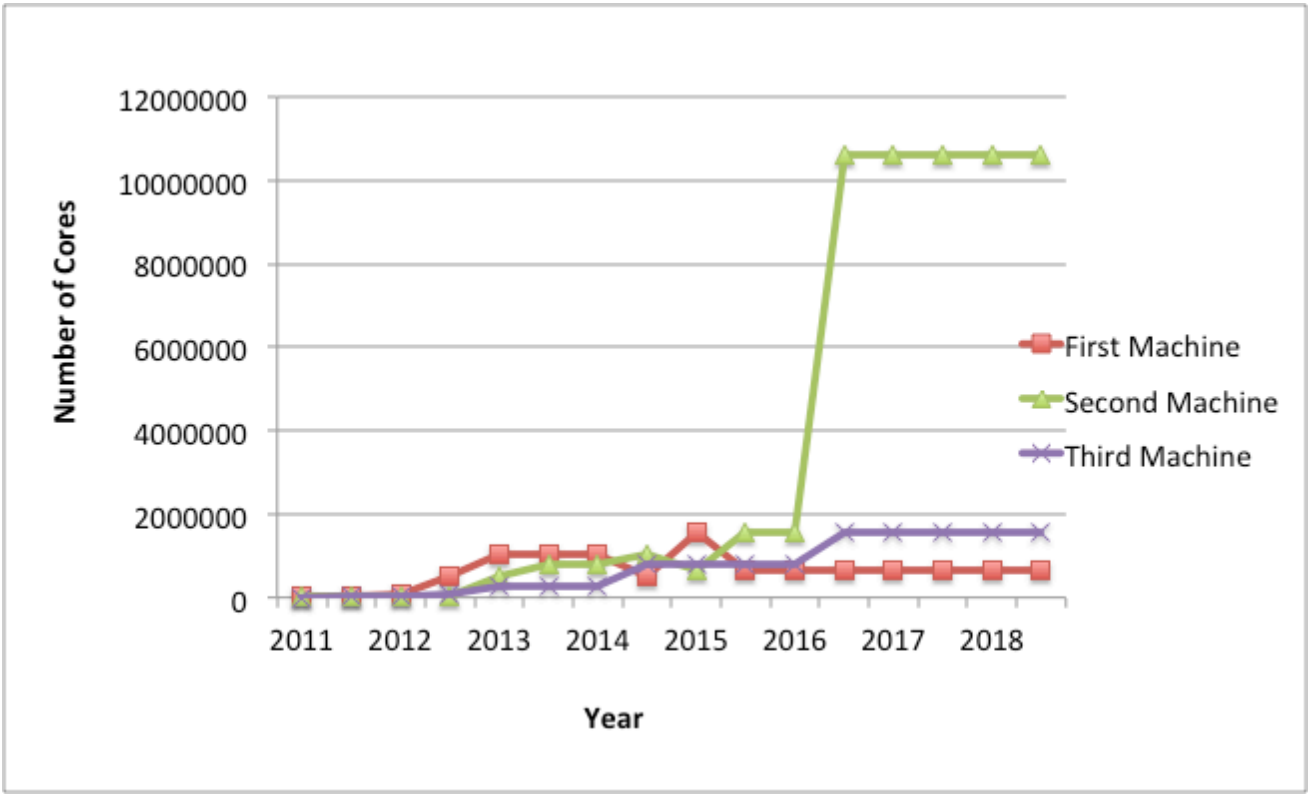


Figure AB-2 Number of Cores Used in Top Three Graph 500 Machines Since Inception of Graph 500 Competition

4.1.2. TECHNOLOGY NEEDS

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. While we expect this factor 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 the memory and the global network. 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.

To increase performance, memory bandwidth needs to increase. The current top machines have an aggregate memory bandwidth of about 5 petabytes per second. We need to increase this number. We expect that the next generation machines will attain about 20 petabytes per second, thanks to in-package dynamic random access memories (DRAMs) such as the high-bandwidth memory (HBM) standard. This will result in a noticeable performance improvement.

To further improve performance, global networks need to provide higher bandwidths and lower latencies. What is especially needed is a higher injection bandwidth into the network, as well as faster all-to-all communication support. The current top machines already have very expensive networks; we expect the cost of such networks to increase in the future, as they are upgraded. One possible source of improvements is to use optical networks, especially if they can provide aggregate injection rates into the network on the order of one terabit per second.

4.1.3. SYSTEMS AND ARCHITECTURES IMPACT

The 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.

In terms of algorithms, 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. We also expect the benchmarks to change to include benchmarks that modify the graph on the fly.

In these clusters, the cores can be out-of-order commodity cores, with modest floating-point support. Since there is a large frequency 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) can 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.

Memory bandwidth needs to be high. In the next few years, we expect to see an aggregate memory bandwidth of about 10 petabytes per second, thanks to the arrival of stacked in-package memory such as HBM. This will deliver a good performance boost, at the expense of increasing 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.

The highest gains will likely be obtained by improving the global network bandwidth and latency. We expect to see more custom-designed networks and possibly optics-based networks, hopefully soon delivering an aggregate bandwidth into the network of around one terabit per second. These networks need to provide fast all-to-all communication, so that cores can exchange boundary information quickly. They need to support irregular communication patterns. Some form of network topology that handles non-local traffic effectively will work best.

4.2. Feature Recognition

The application area of “Feature Recognition” refers to both the training of DNNs and the use (e.g., forward inference) of already-trained DNNs, as well as any other systems that derive their performance characteristics by learning over large sets of data. These benchmarks are designed to track progress in this field and enable continued evolutionary improvement of existing computational approaches (e.g., CPUs and GPUs), as well as to provide accurate performance targets for evolutionary approaches such as approximate [5] and systolic [6] digital computing techniques, or for revolutionary approaches such as crossbar arrays of analog memory devices [7].

Metrics that are relevant include both computational capability (speed) and energy efficiency in doing these computations. Computational capability is most readily available in terms of raw compute capability (tera(FL)OP/second). Note that sometimes operations are performed on data-words with many fewer bits than the 32-bit single-precision, necessitating the use of teraOP rather than teraFLOP. Energy efficiency of hardware can be computed in terms of tera(FL)OP/second/W (equivalent to tera(FL)OP/joule).

Unfortunately, the actual throughput in terms of examples either inferenced or trained per second can depend on more than just raw compute capability, including memory bandwidth and network delays. Network delays can be incurred waiting for other training hardware in distributed training, or in waiting for a sufficiently large test-batch to be assembled in an inference environment where a large batch-size was chosen for optimal compute efficiency.

A significant challenge here is how to accurately distinguish performance and efficiency improvements that stem from improvements in the underlying hardware from improvements due to algorithmic improvements (like weight pruning, compression, and reduced precision), yet mandate an acceptable classification accuracy. Non-hardware-based improvements are of course eagerly welcomed, but the significant presence of such improvements can make accurate benchmarking considerably more challenging

This application area breaks naturally into two main components:

1. Forward-evaluation (also known as forward-inference) of already trained DNNs. Throughput is quantified in units of classification-per-second, and efficiency in units of classifications-per-second-per-watt. Much of this computation might take place in mobile or other power-constrained platforms, so energy efficiency is quite important. In a server environment, latency (in delivering inferences) is important [8], which tends to favor lower batch-sizes.

2. Training of the weights (and any other adjustable parameters) for DNNs. Metrics here are time-to-complete training for a given accuracy, as well as time-and-energy-to-complete-training.

There are a number of other challenges associated with benchmarking this application area. It is common practice to consider DNNs to be "fully trained" despite the fact that the classification accuracies on previously unseen examples (e.g., generalization accuracy) will remain well below 100%. As a result, it can be difficult to define the "completion" of training.

Furthermore, no quantitative metric for accuracy on any given dataset will remain relevant for very long, due to the rapid and continued evolution of the DNN field. Worse yet, datasets that were once highly relevant, competitive and useful can become useless for benchmarking due to algorithmic improvements that render that dataset "too easy." Achieving an "expected" accuracy will only become more important in the future, since emerging approaches for speeding up both the forward-evaluation and the training of DNN involve approximations: either reduced-precision (all the way down to 1–2 bit weights to reduce memory footprint for forward evaluate), weight pruning and compression, or analog techniques.

There are numerous ways in which time-per-classification or time-to-complete-training can improve, which complicates benchmarking considerably. For example, changes in software framework can change the time-per-training-example on the same hardware by a factor of 20 [9]. As a result, benchmark numbers for two different hardware platforms can differ because the underlying hardware is different, or because the software frameworks used were different.

While there are some emerging benchmarks that focus on essential component blocks of DNN systems [10], 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.

4.2.1. ISSUES COMPLICATING BENCHMARKING

An ideal benchmarking approach would be to benchmark one "well-known" example network for each of the different types of networks expected to be a future importance. These would include convolutional neural networks (CNN) for image processing, fully-connected deep neural networks (FC-DNN) for audio/speech processing, long short-term memory (LSTM) for natural language processing, and other candidates as needed.

By using well-known and defined example networks (in terms of number of layers, number of neurons and synapses, the addons such as dropout and momentum that were used when the network was first published), we would be able to specify that any alternative approach (pruning weights, reduced precision, analog approaches, etc.) must achieve a classification accuracy that is within the run-to-run standard deviation of a conventional implementation of the full network. This could then provide an accuracy target, at which the time-for-training, time-and-energy-for-training, time-for-classifying-an-example, and time-and-power-for-classifying-an-example could each be quantified.

The problem is that such an approach is difficult to enforce over long periods of time, as early hardware struggles with networks that are nearly trivial for later hardware. This is particularly true given that while raw TFLOPs strongly influences the wall-clock performance for CNN networks[8], execution time for networks with many fewer operations per weight element tends to depend more heavily on bandwidth into the CPU or GPU, particularly memory bandwidth. For such networks, it becomes challenging to prevent hardware with large computational capabilities from being idle most of the time while waiting for data (particularly for weight data). While neuromorphic approaches could potentially perform computations at the location of weight data[7], it remains an open question whether such approaches can deliver equivalent accuracy for training and inference of DNN networks of interest.

4.2.2. MLPERF.ORG

In May 2018, a consortium of companies and universities announced [11] the creation of the ***MLperf.org*** website and github repository. Currently grown to 28 companies and 7 universities, the consortium's website describes a broad benchmark suite for machine learning (ML) applications. Motivated by the SPEC benchmark, "The MLPerf effort aims to build a common set of benchmarks that enables the ML field to measure system performance for both training and inference from mobile devices to cloud services. We believe that a widely accepted benchmark suite will benefit the entire community, including researchers, developers, builders of machine learning frameworks, cloud service providers, hardware manufacturers, application providers, and end users.

Learning from the 40-year history of benchmarks, MLPerf has these primary goals:

1. Accelerate progress in ML via fair and useful measurement
2. Serve both the commercial and research communities

3. Enable fair comparison of competing systems yet encourage innovation to improve the state-of-the-art of ML
4. Enforce replicability to ensure reliable results
5. Keep benchmarking effort affordable so all can participate" [12]

Two divisions—"Closed" and "Open"—appear designed to address some of the considerations mentioned in previous sections, in terms of distinguishing performance improvements that are truly due to hardware while also encouraging algorithmic, framework, and other software improvements.

While there is language on the site that discusses inference, the primary metric is described as "wall clock time to train a model to a target quality" which would only be applicable for training. There is also a mention of power and cost to be part of future benchmarking results.

Standard reference datasets are provided for 7 benchmarks [13]:

1. **image_classification**—Resnet-50 v1 applied to Imagenet.
2. **object_detection**—Mask R-CNN applied to COCO.
3. **single_stage_detector**—SSD applied to COCO 2017.
4. **speech_recognition**—DeepSpeech2 applied to Librispeech.
5. **translation**—Transformer applied to WMT English-German.
6. **recommendation**—Neural Collaborative Filtering applied to MovieLens 20 Million (ml-20m).
7. **sentiment_analysis**—Seq-CNN applied to IMDB dataset.

As of this writing (early November 2018), however, no public benchmarking results are yet posted for these reference training datasets, although there is a Nov 9 submission date for entries that could indicate this will change in the near future. Datasets and reference-tasks have apparently not yet been finalized for the inference benchmarking component of ***MLperf.org***.

4.2.3. 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.

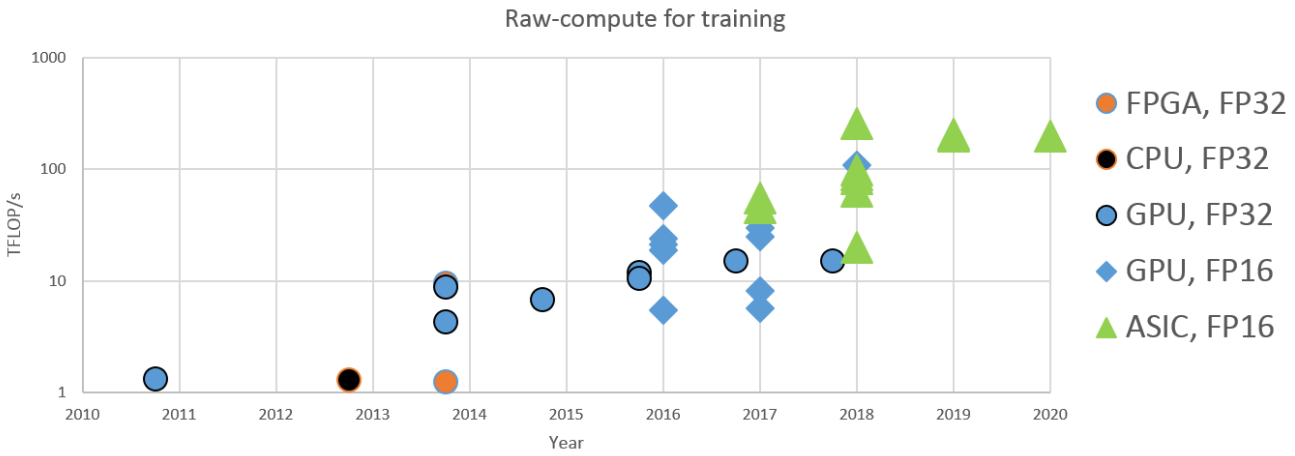


Figure AB-3 Raw Compute Capability of Existing and Projected Hardware for Training, as per Table AB-3

Figure AB-3 shows a plot of raw computational capabilities in units of teraFLOP per second for a few recent CPUs and GPUs and projected or advertised capabilities for some near-future DNN hardware (labelled as ASICs). Table AB-3 shows a breakdown of numbers used to generate this plot, including references for where each data point was obtained.

Figure AB-4 shows the companion plot of computational efficiency for the same hardware, in unites of teraFLOP/second/watt. A trendline of 1.9x/year is included to guide the eye only.

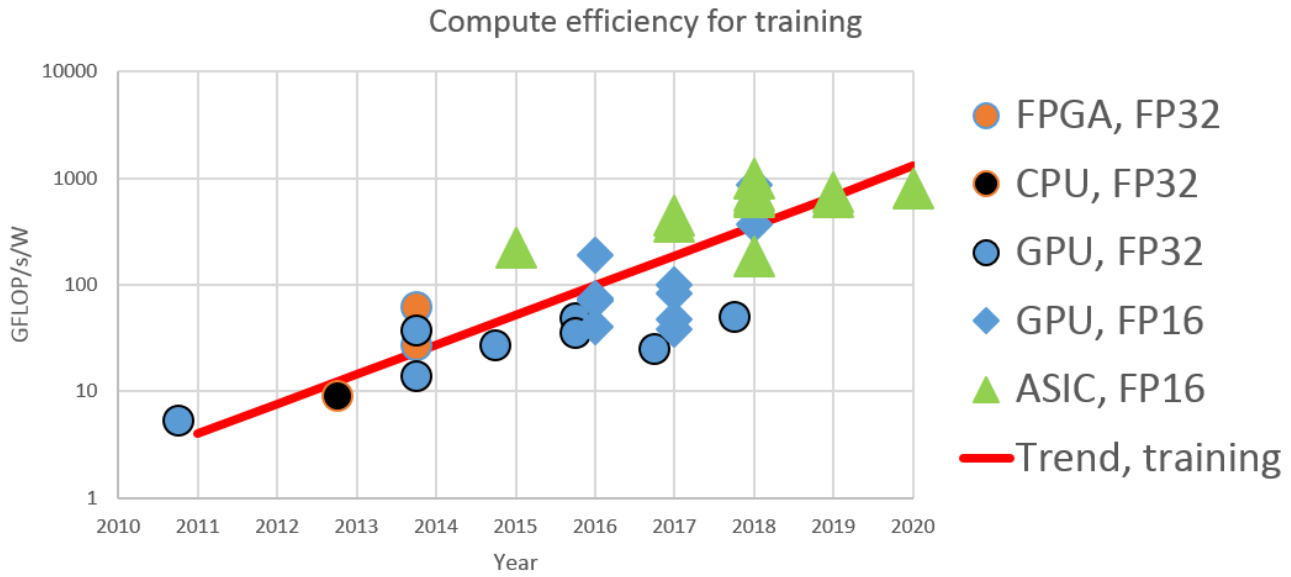


Figure AB-4 Compute Efficiency of Existing and Projected Hardware for Training, as per Table AB-3

Note: A trendline (1.9× per year) is included to guide the eye.

Figure AB-5 shows a similar plot of raw computational capabilities in units of teraOP/second for numerous recent CPUs and GPUs as well as projected or advertised capabilities for some near-future DNN hardware explicitly designed for inference and labelled as tensor processing unit (TPU) or ASICs. Figure AB-6 shows the companion plot of computational efficiency for the same hardware, in units of teraOP/second/watt. A trendline of 1.9× per year is included to guide the eye only. Note that in both plots, but more clearly in the inference hardware, one-time improvements based on reduced precision (from 32 to 16 to 8 or even 6 bits) are clear to see in the data.

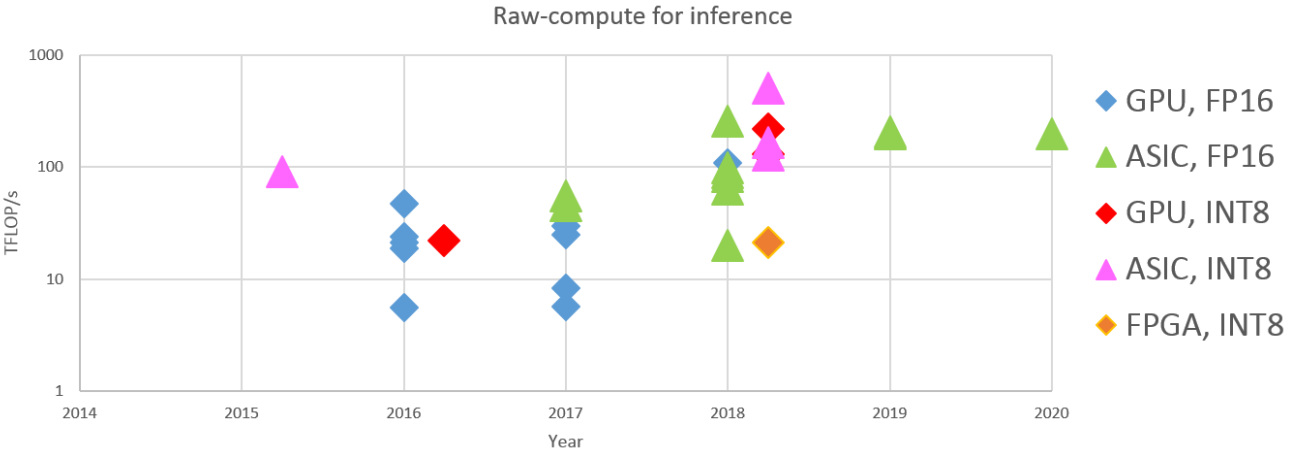


Figure AB-5 Raw Compute Capability of Existing and Projected Hardware for Inference, as per Table AB-3

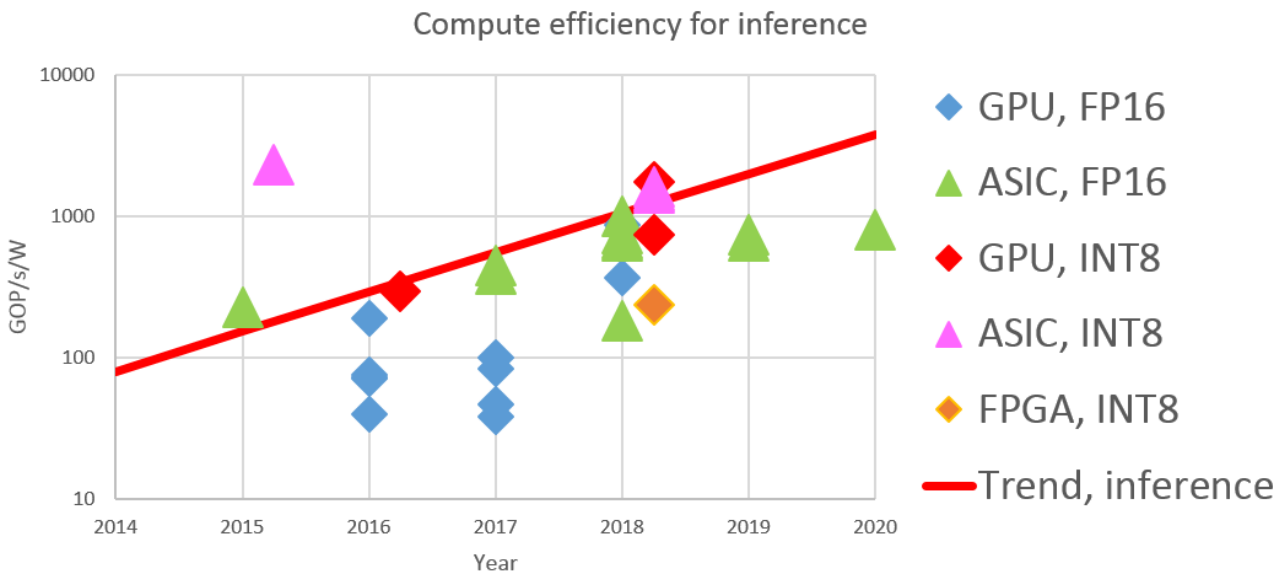


Figure AB-6 Compute Efficiency of Existing and Projected Hardware for Inference, as per Table AB-3

Note: A trendline (1.9x per year) is included to guide the eye.

Table AB-3 Breakdown of numbers used to generate Figures AB-3 through AB-6, including references for where each data point was obtained

	YEAR	INT6 GEMM (TOP/sec)	INT6 GEMM GOP/s/W	FP32 GEMM (TFLOP/s)	FP32 GEMM (GFLOP/s/W)	INT4 compute	INT4 compute (GOP/W/s)	INT8 compute (TOP/sec)	INT8 compute	FP16 compute (TOP/sec)	FP16 compute (GOP/s/W)	FP32 compute (TFLOP/sec)	FP32 compute (GFLOP/s/W)	References
Arria10 FPGA	2014	8	170	1.25	27									[1],[2]
Stratix10 FPGA	2014	70	385	9.5	62									[1],[2]
Xilinx Virtex	2018							21	233.33					[23]
Haswell CPU (E5-2699 v3)	2013							2.6 (per die)	18			1.3 (per die)	9	[3]
Nvidia 2090	2011											1.33	5.3	[4]
Nvidia K40 (Kepler)	2014											4.29	14	[4]
Nvidia K80 (Kepler)	2014											8.74	37	[4]
Nvidia M40 (Maxwell)	2015											6.8	27	[5],[6]
Nvidia P4 (Pascal)	2016							22	293	5.5	73			[6],[7]
Nvidia P40 (Pascal)	2016									47	188	12	48	[6],[8]
Nvidia P100 (Pascal)	2016									24	40	10.6	35	[4],[6]
Nvidia V100 (Volta)	2017									30	100	15	25	[4],[6]
Tesla P100 (NVlink)- Pascal	2016									21.216	70.72			[9]
Tesla P100 (PCIe) -Pascal	2016									18.68	74.72			[9]
AMD MI6	2017									5.7	38			[10]
AMD MI8	2017									8.2	46.86			[10]
AMD Vega	2017									25	83.33			[11]
Nvidia Turing	2018							220	733.33	110	366.67	15	50.00	[17]
Nvidia T4	2018					260	3466.67	130	1733.33	65	866.67			[18]
Google TPU1	2015							92	2300					[3]
Google TPU2	2017									45	225.00			[19,20]
Google TPU3	2018									90	450.00			[19,20]
Intel LakeCrest	2017									55	183.33			[13]
GPU_10nm ("Navi")	2018									98	392			[14]
Nvidia Xavier	2018									20	1000			[15]
GPU_7nm -Ampere	2019									208	693.33			
Intel KnightCrest	2020									200	666.7			
Cambricon MLU-100	2018							128	1600	64	800			[16]
Cambricon MLU-100 (1.3GHz)	2018							166.4	1512.73	83.2	756.36			[16]
GraphCore Colossus	2019									200	666.67			[21]
Huawei Ascend	2018							512	1462.86	256	731.43			[22]

References:

[1] www.eetimes.com/document.asp?doc_id=1317674
[2] www.nextplatform.com/2017/03/21/can-fpgas-beat-gpus-accelerating-next-generation-deep-learning/
[3] arxiv.org/abs/1704.04760
[4] wccftech.com/nvidia-volta-gv100-gpu-tesla-v100-architecture-specifications-deep-dive/
[5] images.nvidia.com/content/tesla/pdf/nvidia-tesla-m4-datasheet.pdf
[6] www.anandtech.com/show/10433/nvidia-announces-pci-express-tesla-p100
[7] images.nvidia.com/content/pdf/tesla/184457-Tesla-P4-Datasheet-NV-Final-Letter-Web.pdf
[8] images.nvidia.com/content/pdf/tesla/184427-Tesla-P40-Datasheet-NV-Final-Letter-Web.pdf
[9] en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units
[10] www.extremetech.com/computing/240908-amd-launches-new-radeon-instinct-gpus-tackle-deep-learning-artificial-intelligence
[11] hothardware.com/news/amd-radeon-rx-vega-unveiled-13-tflops-16gb-hbm2-4k60
[12] www.nextplatform.com/2017/05/22/hood-googles-tpu2-machine-learning-clusters/
[13] syncedreview.com/2017/04/15/what-does-it-take-for-intel-to-seize-the-ai-market/
[14] www.tweaktown.com/news/55875/amd-launch-monster-navi-10-2019-next-gen-ram/index.html
[15] wccftech.com/nvidia-xavier-soc-tegra-volta-gpu-announced/
[16] www.anandtech.com/show/12815/cambricon-makers-of-huaweis-kirin-npu-ip-build-a-big-ai-chip-and-pcie-card
[17] hothardware.com/news/nvidia-geforce-rtx-1080-rtx-1080-ti-799-1199-september-20th
[18] www.tomshardware.com/news/nvidia-tesla-t4-turing-gpu,37788.html
[19] www.zdnet.com/article/gpu-killer-google-reveals-just-how-powerful-its-tpu2-chip-really-is/
[20] www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/
[21] supercomputersfordl2017.github.io/Presentations/SimonKnowlesGraphCore.pdf
[22] mybroadband.co.za/news/technology/279227-huawei-ascend-910-ai-chip-unveiled-the-greatest-computing-density-on-a-single-chip.html
[23] 2.11_Xilinx_HC30_Xilinx_Rahul_Nimaiyar_08212018_v2.pdf, HotChips 2018

**References for Table AB-3 are also found with the spreadsheet.*

Table AB-4 Comparison of expected and reported inference compute-capability (orange, in images/second) and inference compute efficiency (green, in images/W/second) for several existing GPUs, as per Table AB-3.

		Specifications		Expected inference		Reported inference		Fraction of expected performance achieved	
				AlexNet (0.724 GFLOP/image)	GoogLeNet (1.43 GFLOP/image)	AlexNet	GoogLeNet	AlexNet	GoogLeNet
Nvidia M4 (Maxwell)	FP32	2.2 TFLOP/s	29 GFLOP/s/W	40.06	20.28	33 images/W/sec	12 images/W/sec	82.4%	59.2%
Nvidia M40 (Maxwell)	FP32	6.8 TFLOP/s	27 GFLOP/s/W	9392	4755	3200 images/sec	1446 images/sec	34.1%	30.4%
Nvidia P4 (Pascal)	INT8	22 TOP/s	293 GOP/s/W	404.70	204.90	169 images/W/sec	91 images/W/sec	41.8%	44.4%
Nvidia P40 (Pascal)	INT8	47 TOP/s	188 GOP/s/W	64917	32867	16280 images/sec	6408 images/sec	25.1%	19.5%
	FP32	12 TFLOP/s	48 GFLOP/s/W	16575	8392	5198 images/sec	2215 images/sec	31.4%	26.4%

**References for Table AB-4 are also found with the spreadsheet.*

Table AB-3 compares the inference compute-capabilities (shown in orange, in units of images/second) and inference compute-efficiency (green, images/joule) that should be expected given the computation required and the product specifications against the actual reported inference computational performance and/or efficiency, for several recent GPUs. The two networks shown here are convolutional neural networks, which require the most operations per weight data element received, and thus which ought to offer the highest potential computational efficiencies. Despite this, the actual performance or efficiency achieved is typically well below what would be expected, sometimes by a factor of 5×. As a result, other networks that will perform fewer operations on each weight data element – such as fully-connected networks or long short-term memory – can be expected to have the computational hardware idle even more often, and thus fall even further below the expected performance based on the rated teraOP/second.

4.2.4. TECHNOLOGY NEEDS

For near-term digital hardware, the critical hardware needs are the design of systolic computing units that align well with the DNN algorithms [6,8] and being able to receive and send data to large amounts of memory at high-bandwidth yet reasonable power. Further improvements can be obtained for distributed training by highly efficient use of network bandwidth, allowing the multiple "workers" to collaborate with a minimal amount of information exchange.

For farther-term hardware based on either in-memory digital [14] or analog [7] computation, critical technology needs are analog memory devices offering small yet symmetric changes in conductance over a wide dynamic range; low-power, high-bandwidth, moderate-precision but extremely area-efficient analog-to-digital converters, or other circuit techniques that provide similar parallel communication of neuron activations; and finally, proof that the requisite training accuracies for inference and training can be supported by such in-memory DNN hardware.

4.2.5. SYSTEM AND ARCHITECTURES IMPACT

Architectures that are widely used for the Feature Recognition application area today are GPUs. However, GPUs are not energy efficient due to (perhaps ironically) the bloated microarchitectures that have been expanding over time in an effort by the industry to make GPUs more general-purpose compute devices. In fact, hardware built on large collections of relatively simple digital processing units would be far more efficient.

Fixed-function acceleration plays a significant role in accelerating this application area. Crossbar architectures and related and in-memory computing based on digital or analog memory arrays have been shown to efficiently perform multiply-accumulate computations in place. Memory bandwidth is a critical need for this application space as well. If a sparse implementation of inferencing is used, then memory latency is also critical. Emerging processing near memory could improve this by placing memory and logic together. Similarly, the emergence of reconfigurable hardware accelerators as top performers for inferencing suggests that combined microprocessor and FPGA architectures could improve feature recognition.

4.3. DISCRETE EVENT SIMULATION

A discrete event simulation (DES) models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system [15]. Between consecutive events, the simulation can directly jump in time from one event to the next. In contrast, with continuous simulation, time ins broken up into small time slices and every time slice is simulated to update the system state. DES can typically run much faster than continuous simulation as it does not have to simulate every time slice.

Common uses of DES include diagnosing process issues, testing performance improvement ideas, evaluating capital investment decisions (with Monte Carlo), and simulating computer networks. DES typically include the following components: State, Clock, List of events, Random-number generators, and Statistics.

471.omnetpp in the SPEC CPU 2006 benchmark suite represents a DES workload [16]. This benchmark performs discrete event simulation of a large ethernet network. The simulation is based on the OMNeT++ discrete event simulation system, a generic and open simulation framework [17]. OMNeT++'s primary application area is the simulation of communication networks, but its generic and flexible architecture allows for its use in other areas such as the simulation of IT systems, queueing networks, hardware architectures or business processes as well.

Galois also includes a benchmark that represents discrete event simulation [18]. Two different implementations, based on ordered and unordered algorithms, are present in the Galois suite. Unlike 471.omnetpp in SPEC CPU 2006, the Galois implementations are multi-threaded, although their initial scalability analysis indicates that high number of threads is limited due to lack of parallelism in the workload. Threads have to wait more often to remove items from the worklist, which becomes a bottleneck.

4.3.1. PERFORMANCE TRENDS AND PREDICTION

In this section, we analyze the performance trend of 471.omnetpp using historical data points uploaded to the SPEC database. We downloaded roughly 9,500 data points of various systems running the workload ranging from March 2006 to December 2017. SPEC has stopped accepting new results starting in 2018. We bucketed these data into monthly bins and calculate the maximum and average scores for each bin. SPEC reports the "base" and "peak" scores for each workload. According to SPEC, the base metrics are required for all reported results and have stricter guidelines for compilation. For example, the same flags must be used in the same order for all benchmarks of a given language. The peak metrics are optional and have less strict requirements. For example, different compiler options may be used on each benchmark, and feedback-directed optimization is allowed.

Figure AB-7 shows the performance of 471.omnetpp over time. We plot both base and peak performance with maximum and average scores per monthly bins, represented by the four solid lines. We also plot the linear regression of each metric, represented by the four dotted lines. In general, performance appears to be improving over time. Note that there are occasional steep jumps in performance. These generally align well with major CPU releases as depicted in the figure. One thing to note is that the data uploaded to SPEC only has the test date associated with the data, not the system release date. While, in theory, it is possible to look up all the system release dates for all 9,500 data points, we assumed that, in general, newer systems would be more popular to benchmark at any given time, and, thus, the test dates would align well with the system release dates.

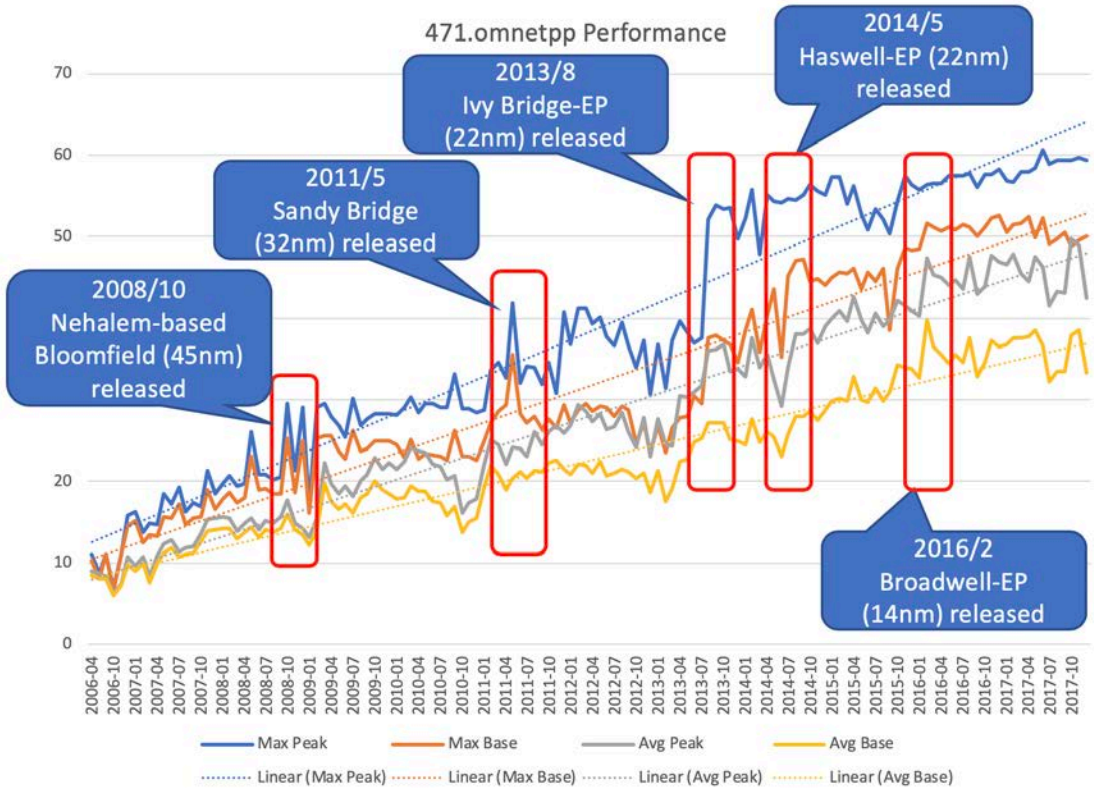


Figure AB-7 Historical Performance of 471.omnetpp Over Time

4.3.2. TECHNOLOGY NEEDS

DES, represented by 471.omnetpp in SPEC CPU 2006, generally has lower than average instructions-per-cycle (IPC). It is characterized to be memory bound with high L2 cache miss rates. It also has a relatively large code footprint and, thus, is sensitive to the instruction cache size.

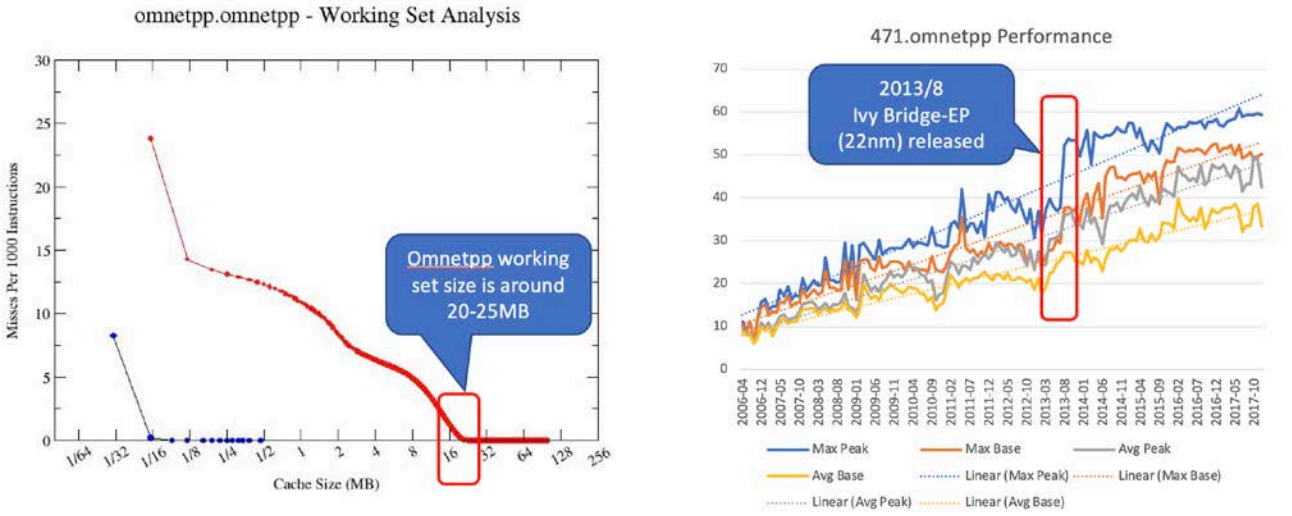


Figure AB-8 Working Set Analysis of 471.omnetpp and Dramatic Jump in Peak Performance Due to Working Set Fitting in Cache

Figure AB-8 (left) shows the working set analysis of 471.omnetpp from [19]. Once the data cache size reaches 20–25 MB, the working set almost completely fits in the cache and the misses per kilo-instructions (MPKI) reaches zero. This could explain the largest jump in 471.omnetpp performance around August of 2013. The Ivy Bridge-EP processor released around this timeframe is the first processor to have a 25 MB last-level cache, which is enough to fit the entire working set of 471.omnetpp. This could also explain why the max peak performance of the benchmark has remained somewhat flat after that point. Of course, the whole domain of DES should not be restricted to this single benchmark with a limited size input set. However, being memory bound and cache sensitive would be a general characteristic to describe DES workloads.

471.omnetpp is also notorious for having an irregular memory access pattern and thus being hard to prefetch. The event queues in DES workloads are typically modeled as priority queues sorted by event time. Regardless of the order in which events are added to the event set, they are removed in strictly chronological order. To improve performance, priority queues typically use a heap as their backbone, giving better performance for inserts and removals. From a computational-complexity standpoint, priority queues are similar to sorting algorithms. Sorting algorithms can exhibit an irregular memory access pattern depending on the input data. Intelligent prefetching and cache replacement to reduce the effective memory access latencies can improve the performance of DES workloads.

4.3.3. System and Architectures Impact

Traditional general-purpose CPUs would still be suited to run DES workloads efficiently. It is unclear whether any type of accelerators can be helpful. Being memory-bound, process-in-memory systems might be interesting, but the need for frequent synchronization would limit the scalability as shown by the Galois benchmarks mentioned earlier.

Most DES workloads would benefit from larger and faster caches. Novel prefetching schemes that can effectively capture the irregularity in the access pattern would significantly help with further improving the performance.

Larger-scale DES workloads would require breaking up the physical system into multiple logical processes. However, this is complicated as the computing nodes will be exchanging timestamped messages while often operating at different points of simulation time. Highly efficient synchronization methods must be used to effectively scale such workloads.

4.4. 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 [20]. The relevant application codes are typically based on finite-element algorithms, while the

computations are fairly heavy workloads that conventionally are dominated by floating-point arithmetic. Examples include application areas such as climate modeling, plasma physics (fusion), medical imaging, fluid flow, and thermo-evolution.

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 [21]. All are realistic kernels, although the authors' claim that they include three "simulated applications" as a more accurate description 25 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 [22] 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 [23]. 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 [24]. 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 [25]. The report confirms "presence" of the 13 Dwarfs in 6 broad application domains – embedded computing, gp 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 actually sufficient for the domains described in the "View from Berkeley" Technical Report.

4.4.1. 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 is required to 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.

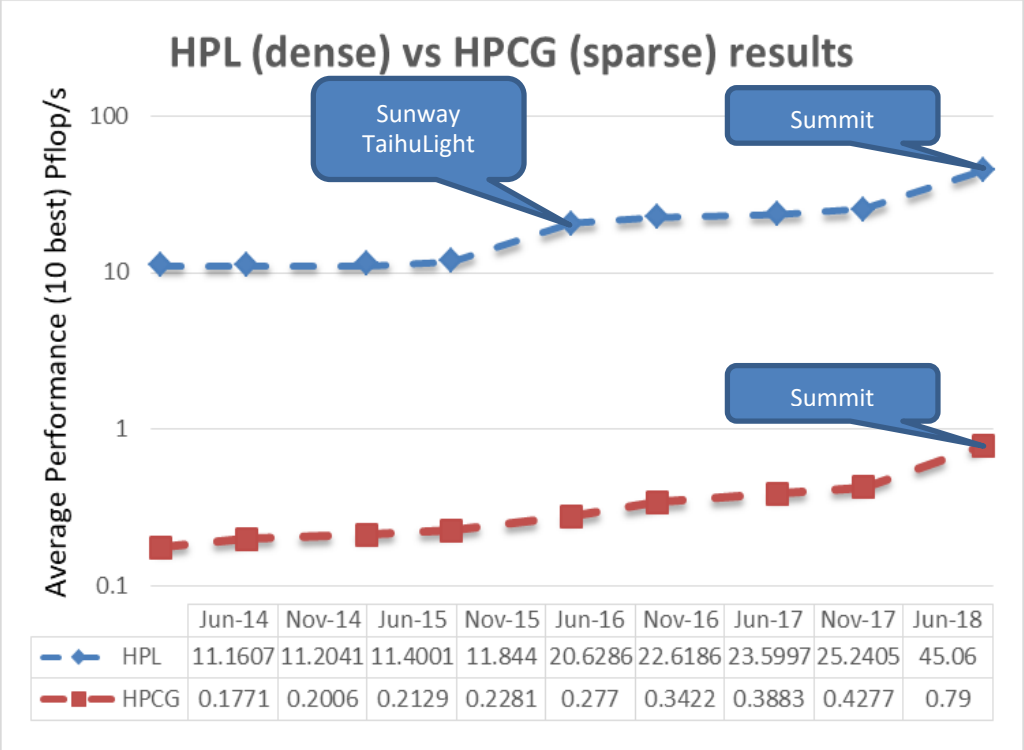
As long as emphasis was on dense problems, this approach resulted in systems with increasing computational performance. This approach was the presumption behind the twenty-five years of semi-annual Top 500 rankings of supercomputers. However, in the last ten years a large number of new applications with very high economic potential have emerged – such as big data analytics, machine learning, real-time feature recognition, recommendation systems, and even physical simulations—that feature irregular or dynamic solution grids. These applications spend much more of their computation in "non-floating point" operations such as address computations and comparisons, with addresses that are no longer very regular or cache-friendly. The "computational intensity" of such programs is far less than for dense kernels, and the complexity of inter-node data transfers can often dominate total compute time. 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 [26].

To highlight the inefficiencies described above, and to identify architectures which may be more efficient, a new benchmark was introduced in 2014 called HPCG[2] (High Performance Conjugate Gradient). HPCG also solves Ax=b problems, but where A is a very sparse matrix[3]. On current systems, floating point efficiency mirrors that seen in full scientific codes. For example, the number 1 supercomputer in the world in terms of dense linear algebra is the Chinese TaihuLight, that same supercomputer is only #5 on HPCG, achieving only 0.4% of its peak floating-point capability on this sparse benchmark. A recent paper [27] analyzed HPCG in detail and came 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.

Therefore, our selected benchmark codes that cover the "Physical System Simulation" application area of interest are the High-Performance LINPACK (HPL) [28][29] and the HPCG [30]. Both are very popular codes with very good regularity of results since June 2014. Another very important reason for selecting HPL and HPCG 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 lay out a relatively well-balanced overall picture of the whole application domain for physical system simulation.

Our approach is to explore a 3-dimensional space – dense systems performance, sparse systems performance, and energy efficiency for both cases. 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.



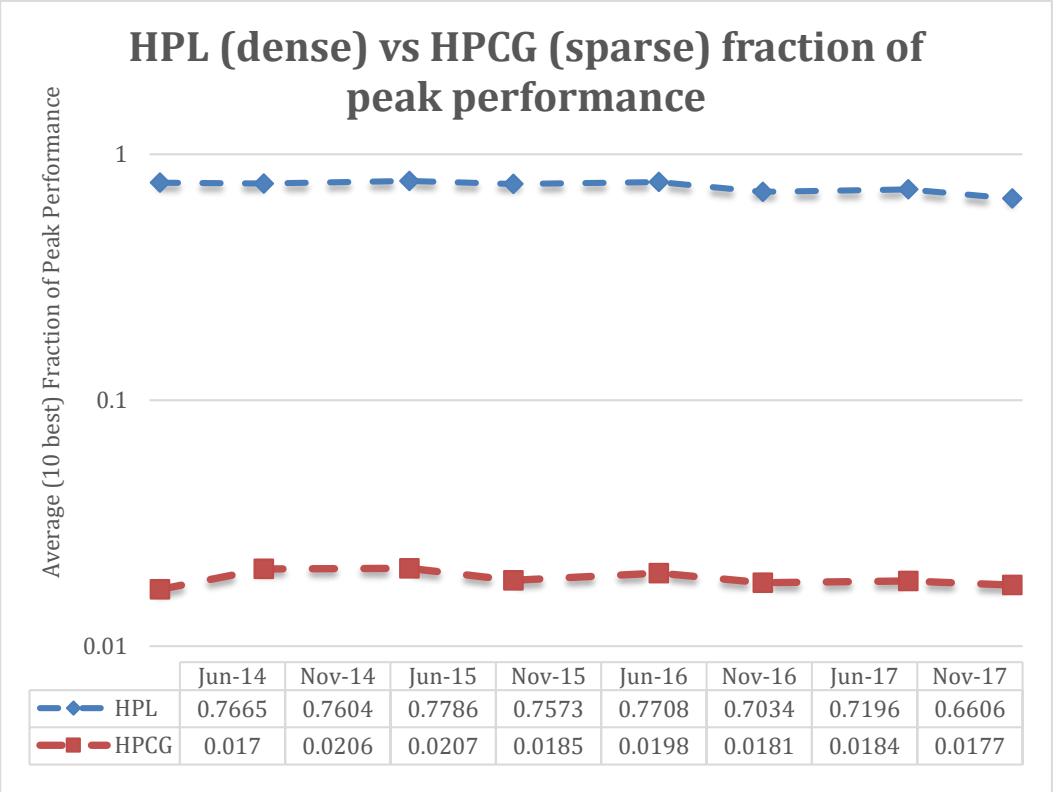
	Jun-14	Nov-14	Jun-15	Nov-15	Jun-16	Nov-16	Jun-17	Nov-17	Jun-18
HPL	11.1607	11.2041	11.4001	11.844	20.6286	22.6186	23.5997	25.2405	45.06
HPCG	0.1771	0.2006	0.2129	0.2281	0.277	0.3422	0.3883	0.4277	0.79

Figure AB-9 HPL (dense systems, blue) vs. HPCG (sparse systems, red) Average Performance

[2] *http://www.hpcg-benchmark.org/*
[3] *Nominally 27 non-zeros in rows of a matrix that may be millions of elements in width.*

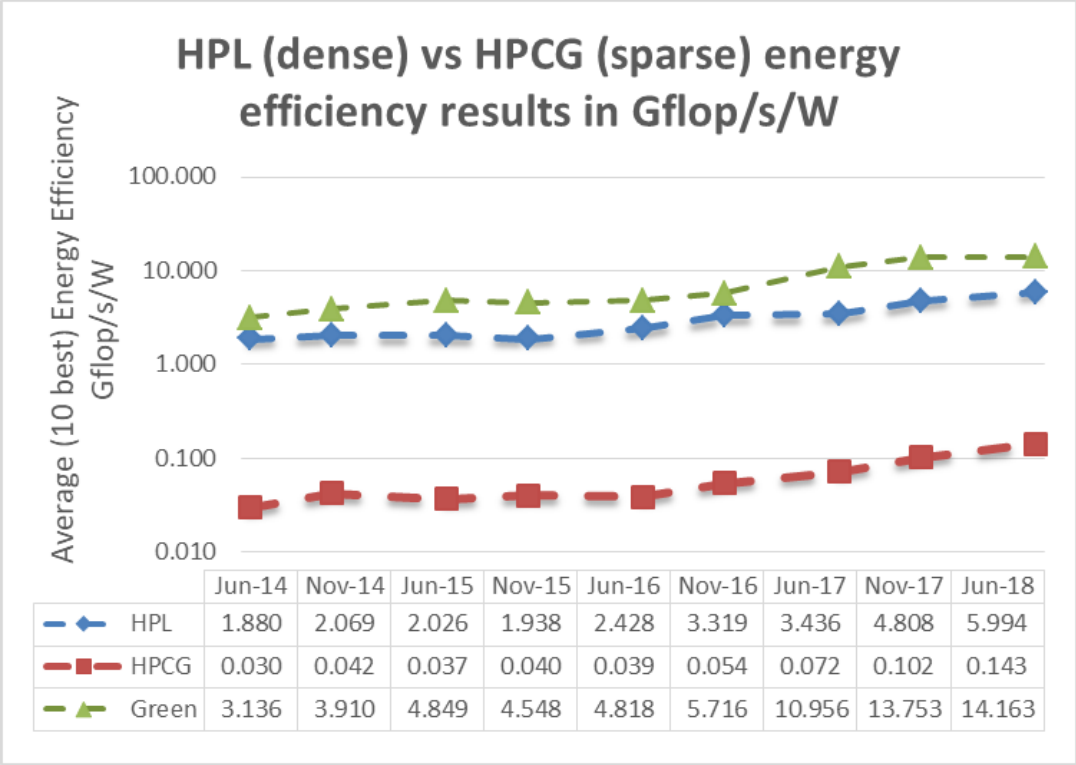
Figure AB-9 shows a significant performance gap of nearly 2 orders of magnitude between HPL and HPCG results in the last several years. The increase of the average HPL performance since June 2016 is because of the introduction of the Chinese Sunway TaihuLight system. The most recent increase of both HPL and HPCG performance is visible since June 2018 after the installation of the Summit supercomputer at ORNL. 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 absolutely clear that achieving Exaflop/s performance with HPL will happen soon, it is equally clear that this achievement will leave this significant gap between dense and sparse system performance!

Figure AB-10 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
HPL	0.7665	0.7604	0.7786	0.7573	0.7708	0.7034	0.7196	0.6606
HPCG	0.017	0.0206	0.0207	0.0185	0.0198	0.0181	0.0184	0.0177

Figure AB-10 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
HPL	1.880	2.069	2.026	1.938	2.428	3.319	3.436	4.808	5.994
HPCG	0.030	0.042	0.037	0.040	0.039	0.054	0.072	0.102	0.143
Green	3.136	3.910	4.849	4.548	4.818	5.716	10.956	13.753	14.163

Figure AB-11 *HPL (dense systems) vs. HPCG (sparse systems) vs. the most energy-efficient supercomputers on the Green 500 list*

As for the energy efficiency dimension (see Figure AB-11), our current designs appear to be in a position to scale up to 200 PetaFLOP/s while remaining below the 20 MW system power consumption requirement. An optimistic estimate based on this would require 5x improvements in energy efficiency, and 10x improvements in the HPL performance currently delivered by the Chinese Sunway TaihuLight system. However, such improvements are not realistic, since the best energy efficiency results and rankings are different from the HPL ranking (see comments above about the top 10 ranked results). Therefore, a more realistic projection based on the Chinese Sunway TaihuLight system is that it needs 10× energy efficiency improvement and 10× higher HPL performance to reach the Exaflop/s barrier. Unfortunately, this would only achieve the desired performance and energy efficiency on 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.

4.4.2. TECHNOLOGY NEEDS

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

a. higher bandwidth and lower latency for accessing and moving data – both locally (memory systems) and remotely (interconnection networks). Breakthrough architecture solutions addressing those challenges could potentially enable up to two orders of magnitude higher performance.

b. Much more efficient implementation of floating-point arithmetic. The IEEE 754-2008 Standard for Floating-Point Arithmetic [31] is due to expire in 2018 unless it is reviewed. However, there does not seem to have been an indication of interest to review it, as of June 2017. While various important aspects of the standard including wasted cycles, energy inefficiencies, and accuracy have been criticized, the path forward is unclear at present. 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 [32].

- More radical approaches proposing new solutions have been under development including the Posit Arithmetic proposal [33].

4.4.3. SYSTEM AND ARCHITECTURES IMPACT

The "Physical System Simulation" application area urgently needs novel and innovative architectures that can help address the 3rd Locality Wall. This includes both novel memory systems and interconnection networks offering much higher bandwidth and lower latency. Energy efficiency indicators need urgent improvements by at least an order of magnitude. This requirement is equally valid for both homogeneous vs. heterogeneous architectures (including accelerators and FPGAs) that need further comparisons and analysis. Since this application area 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 computation.

4.5. OPTIMIZATION

The optimization application area is generally large. We narrow this area to integer-based problems and more specifically to problems that map to the well-known Traveling Salesman Problem (TSP). Of these approaches, there are two different categories: near-optimal techniques (also known as "heuristic techniques"), and (true) optimal techniques. The former is dominated by two classes of algorithms: simulated annealing (SA) and genetic algorithms (GA). (Note that GA is actually a subclass of Evolutionary Algorithms (EA), however it is the most prominent member of EAs.) For true optimal techniques, there are several algorithmic approaches such as dynamic programming (DP), integer linear programming (ILP) and branch-and-bound (BaB). For TSP, each of these approaches can be shown to be NP-hard.

The AB Focus Team found only a few benchmarks that meet the criteria enumerated in Section 1 for true optimal techniques. Thus, for this edition of the roadmap, we focus on near-optimal techniques. For the remainder of these, we assessed several options:

Potential Public Benchmarks

(1) SPEC2k6 [34]: although this solves a vehicle scheduling problem and uses the (accurate, non-NP-hard) network simplex algorithm.

(2) SPEC2k6 [35]: performs a heuristic (A* algorithm) routing of a path through a 2D graph.

(3) PARSEC: canneal is a cache-aware simulated annealing (SA) to minimize the routing cost of a chip design.

We ruled out options (1) and (3) for different reasons. For option (1), 429.mcf is known not for its computational load but for its unpredictable memory behavior. Since the Big Data application area already addresses the latter and 429.mcf's memory stressload and since we do not believe it is typical of the Optimization application area, we chose to not use 429.mcf. For option (3), we could not find sufficient published results for canneal (or any PARSEC benchmark) on real systems. This is because of the intent of PARSEC: to promote research into parallel systems, not to benchmark existing systems. Thus, we chose 473.astar as the benchmark to track for the Optimization application area in this report.

There is not a version of astar included in the latest SPEC CPU2017 benchmark suite. New benchmarks have been added. Deepsjeng, leela, exchange2 model areas of Artificial Intelligence in the form of alpha-beta tree search (Chess), Monte Carlo tree search (Go), and recursive solution generator (Sudoku), respectively. These benchmarks can be broadly classified as optimization algorithms. Going forward, we will track performance of this benchmarks as more results become available.

4.5.1. PERFORMANCE TRENDS AND PREDICTION

In this section, we analyze the performance trend of 473.astar using historical data points uploaded to the SPEC database. This analysis mirrors the analysis of 471.omnetpp above in Section 4.2. We downloaded roughly 9,500 data points of various systems running the workload ranging from April 2006 to December 2017. SPEC has stopped accepting new results in 2018 for this benchmark. We bucketed these data into monthly bins and calculated the maximum and average scores for each bin. SPEC reports the "base" and "peak" scores for each workload. According to SPEC, the base metrics are required for all reported results and have stricter guidelines for compilation. For example, the same flags must be used in the same order for all benchmarks of a given language. The peak metrics are optional and have less strict requirements. For example, different compiler options may be used on each benchmark, and feedback-directed optimization is allowed.

Figure AB-12 shows the performance of 473.astar over time. We plot both base and peak performance with maximum and average scores per monthly bins, represented by the four solid lines. We also plot the linear regression of each metric, represented by the four dotted lines. In general, performance appears to be improving over time. Note that there are occasional steep jumps in performance. As with 471.omnetpp, these generally align with major CPU releases from industry. One thing to note is that the data uploaded to SPEC only has the test date associated with the data, not the system release

date. While, in theory, it is possible to look up all the system release dates for all 9,500 data points, we assumed that, in general, newer systems would be more popular to benchmark at any given time, and, thus, the test dates would align well with the system release dates.

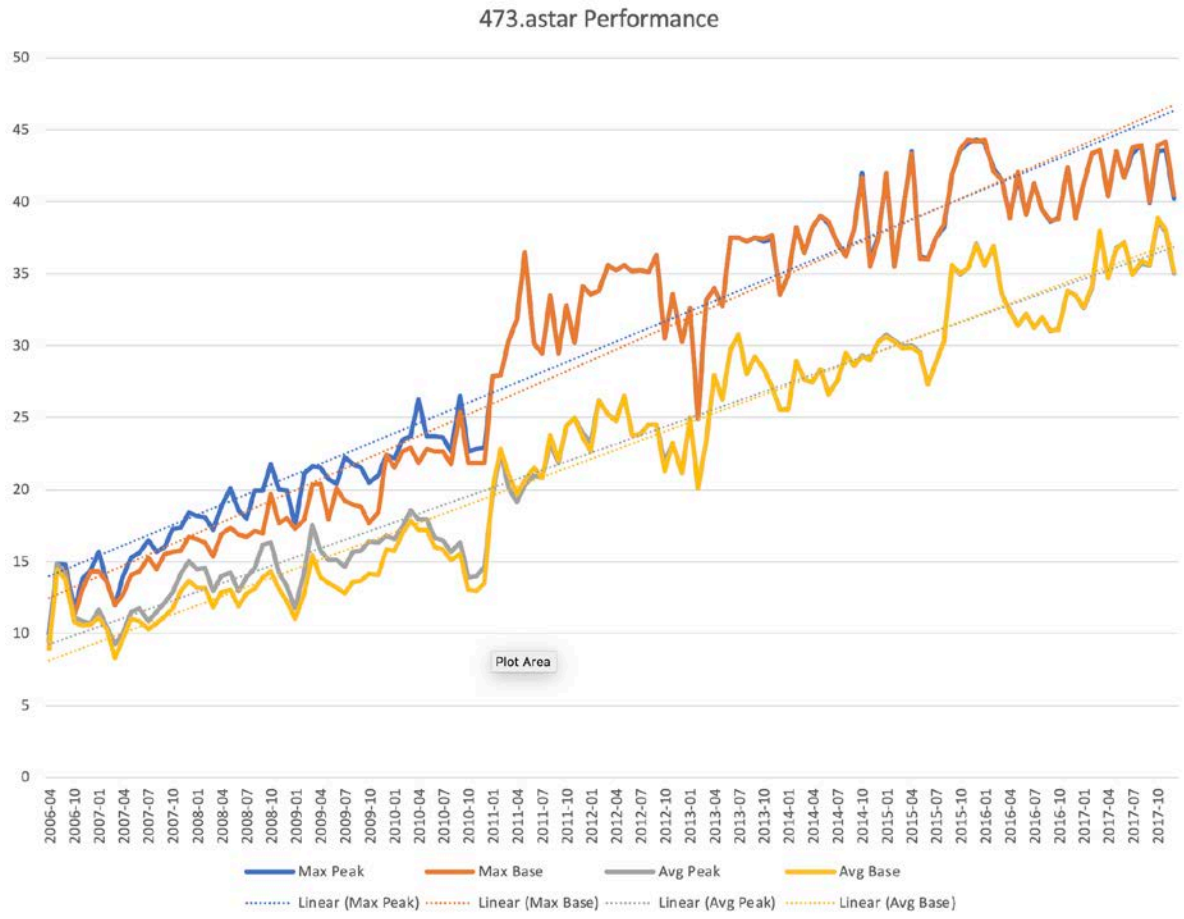


Figure AB-12 Historical Performance of 473.astar Over Time

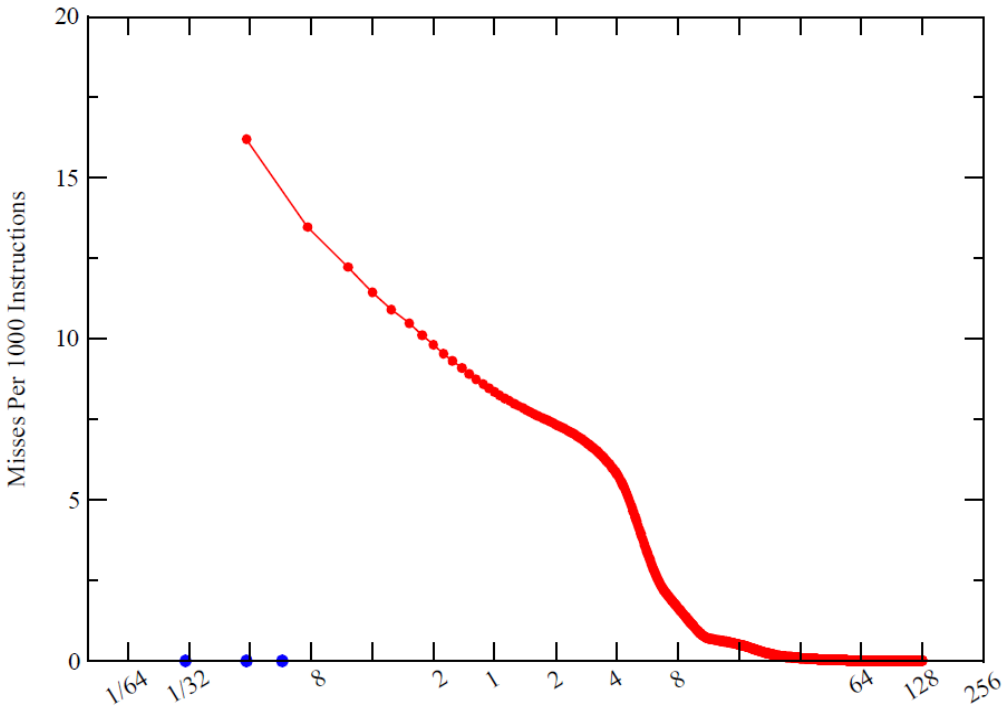


Figure AB-13 Working Set Analysis of 473.astar is Approximately 16–20 MB

Figure AB-13 shows the working set analysis of 473.astar from [36]. Once the data cache size reaches between 16 and 20 MB, the working set almost completely fits in the cache and the misses per kilo-instructions (MPKI) reaches zero. As with omnetpp, 473.astar is also notorious for having an irregular memory access pattern and thus being hard to prefetch.

4.5.2. TECHNOLOGY NEEDS

In general, near-optimal search should be a compute-bound problem. However, and especially in the case of 473.astar, the data used for the optimization (in this case a topological map) is large and very susceptible to cache and memory bandwidth and latency. Thus, for larger optimization problems, the memory subsystem also becomes critical.

4.5.3. SYSTEM AND ARCHITECTURES IMPACT

Optimization in general is an integer core processing problem. When the space being optimized is large, it is constrained by memory. Thus, large HPC-oriented nodes perform best on these problems. We would expect that optimization improves with the improvement in raw compute ability per core.

4.6. GRAPHICS/VR/AR

The AB FT decided that because the best available benchmark that fits the criteria enumerated in Section 1 was the SPECviewperf 12 benchmark from SPEC. SPECviewperf 12 is defined as follows:

> The benchmark measures the 3D graphics performance of systems running under the OpenGL and Direct X application programming interfaces. The benchmark's workloads, called viewsets, represent graphics content and behavior from actual applications [37].

There are three versions of SPECviewperf 12, versions 12.01, 12.02 and 12.1. This relates generally to minor changes and the results across these versions are comparable. One significant change concerns the viewsets. There are nine viewsets: 3dsmax-05, catia-04, creo-01, energy-01, maya-04, medical-01, showcase-01, snx-02 and sw-03. Of these, 3dsmax-05 was added in SPECviewperf 12.1 in 2016. As such, it has a shorter history than the others. We chose to exclude it from our analysis for this reason.

4.6.1. PERFORMANCE TRENDS AND PREDICTION

The SPECviewperf 12 results for viewsets catia-04, creo-01, energy-01, maya-04, medical-01, showcase-01, snx-02 and sw-03 were combined via geometric mean and are shown below in Figure AB-14. Similar to SPECint benchmarks

471.onnetpp and 473.astar, we grouped results into monthly buckets. However, there are less data points for SPECviewperf 12 than for the two SPECint benchmarks. In total there are 119 result spanning February 2014 through September 2017. As a result, the max per bucket geometric mean (Figure AB-14(a)) and the average per bucket geometric mean (Figure AB-14(b)) virtually identical the trend lines.

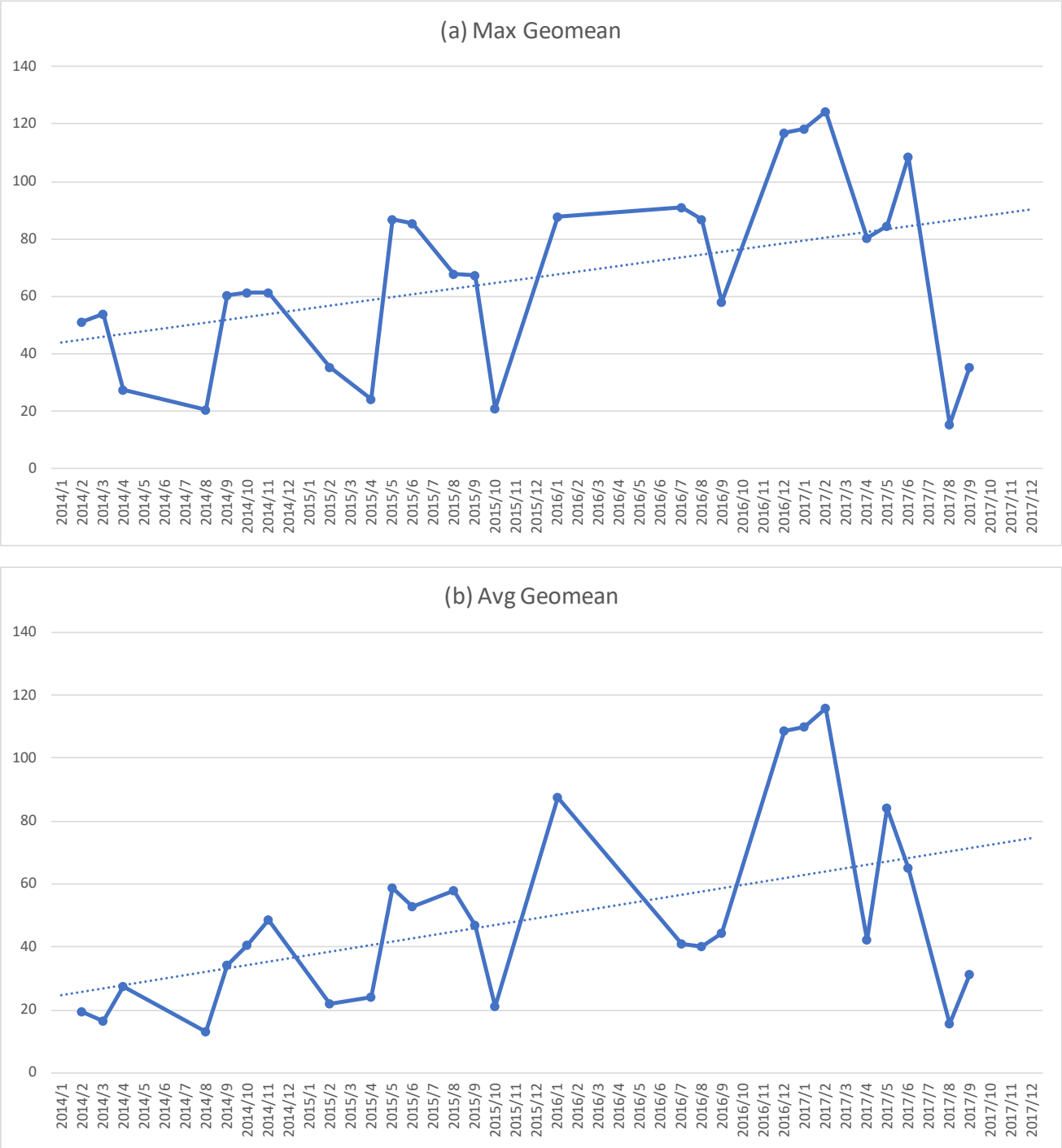


Figure AB-14 Geometric Mean Across Viewsets vs. Time for SPECviewperf 12 Benchmark

Note: Shown is (a) the maximum of the geometric mean for results reported in a given month, and (b) the simple arithmetic average of the monthly geometric means.

The trendline for max geomean shows an increase of approximately 1.4% per month, or 19% per year. There is much noise in the data due to the differences in system classes under test.

4.6.2. TECHNOLOGY NEEDS

Graphics performance depends on many variables such as software framework implementation, algorithm advances, and hardware advances, and SPECviewperf 12 includes all of these. However, a closer inspection of the results shows that there is a correlation between the introduction of new GPU architectures and an increase in results. Generally, GPU architectures improve in two dimensions: the amount of parallelism possible in hardware and the memory bandwidth provided. The former of these two saturates as overall parallelism, although high for graphics applications, is not infinite. The latter is the driver: the higher the memory bandwidth, the better the performance of Graphics/VR/AR workloads.

4.6.3. SYSTEM AND ARCHITECTURES IMPACT

We believe the best architectures for Graphics/VR/AR workloads are highly parallel and have high bandwidth to memory. Because the parallelism levels are high, it is memory bandwidth more than memory access latency that dominates this applications area. Emerging near-memory-processing architectures achieve both high bandwidth and, for well partitioned data, low access latency. Note that existing GPUs grew out of fixed-function accelerators optimized for the (now retired) fixed graphics pipeline. For this reason, this workload is also receptive to fixed-function acceleration.

5. CROSS MATRIX

The function of the Cross Matrix is to map application areas to SA-IFT-defined market drivers. In the May 2016 meeting in Leuven, Belgium, the initial market drivers were determined. In the December 2016 IRDS meeting, the final market drivers for the 2017 Roadmap were decided on. These are:

- **Internet-of-Things edge devices (IoT-e):** IoT is a broad class if computing applications spanning the server to the ultimate sensors and actuators. This market driver focuses on the latter: embedded systems for sensing and actuating, often with tight power constraints.
- **Cloud computing (CC):** This category is for server devices deployed in data centers. This market driver includes both devices designed for accelerating search and for devices designed for high-performance computing (HPC) applications.
- **Cyber-Physical Systems (CPS):** computer-based control of physical devices characterized by real-time processing and used primarily in industrial control. (Germany has an initiative called “Industry 4.0” that, among other things, discusses this interpretation of CPS.)
- **Mobile (Mo):** This category is focused on mobile devices generally, and smartphone devices in particular.

Below in Table AB-4 is the cross matrix that was developed, “X” means important application area(s), and “P” means important and power-constrained application area(s).

Table AB-5 Cross Matrix of Application Area vs. Market Drivers

Application Area	IoT-e	CC	CPS	Mo
Big Data Analytics	P	X		
Feature Recognition	X	X	P	P
Discrete Event Simulation		X		
Physical System Simulation		X		
Optimization		X	P	
Graphics/VR/AR		X		P

6. 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 showed 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 and Graphics/VR/AR.

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.

7. REFERENCES

References for Table AB-3 and Table AB-4:

[1] www.eetimes.com/document.asp?doc_id=1317674

[2] www.nextplatform.com/2017/03/21/can-fpgas-beat-gpus-accelerating-next-generation-deep-learning/

[3] https://arxiv.org/abs/1704.04760

[4] https://wccftech.com/nvidia-volta-gv100-gpu-tesla-v100-architecture-specifications-deep-dive/

[5] http://www.onestopsystems.com/sites/default/files/pdf/nvidia_m4_datasheet.pdf

[6] https://www.anandtech.com/show/10433/nvidia-announces-pci-express-tesla-p100

[7] images.nvidia.com/content/pdf/tesla/184457-Tesla-P4-Datasheet-NV-Final-Letter-Web.pdf

[8] images.nvidia.com/content/pdf/tesla/184427-Tesla-P40-Datasheet-NV-Final-Letter-Web.pdf

[9] en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

[10] www.extremetech.com/computing/240908-amd-launches-new-radeon-instinct-gpus-tackle-deep-learning-artificial-intelligence

[11] hothardware.com/news/amd-radeon-rx-vega-unveiled-13-tflops-16gb-hbm2-4k60

[12] https://www.nextplatform.com/2017/05/22/hood-googles-tpu2-machine-learning-clusters/

[13] syncedreview.com/2017/04/15/what-does-it-take-for-intel-to-seize-the-ai-market/

[14] https://www.tweaktown.com/news/55875/amd-launch-monster-navi-10-2019-next-gen-ram/index.html

[15] wccftech.com/nvidia-xavier-soc-tegra-volta-gpu-announced/

[16] http://www.onestopsystems.com/sites/default/files/pdf/nvidia_m4_datasheet.pdf

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

---

1 http://graph500.org/

2 http://graph500.org/?page_id=446

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 S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision.” In *Proc ICML*, pp. 1737–1746. (2015).

6 Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE J. Solid-State Circ.*, **52**(1):127–138, 2017.

7 P. Narayanan, A. Fumarola, L. L. Sanches, K. Hosokawa, S. C. Lewis, R. M. Shelby, and G. W. Burr. “Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory,” *IBM J. Res. Dev.* **61**(4), 1–11 (2017).

8 ISCA '17 Proceedings of the 44th Annual International Symposium on Computer Architecture, pages 1-12. https://doi.org/10.1145/3140659.3080246

9 https://github.com/soumith/convnet-benchmarks

10 https://svail.github.io/DeepBench/

11 https://www.hpcwire.com/2018/05/02/mlperf-will-new-machine-learning-benchmark-help-propel-ai-forward/

12 http://mlperf.org

13 https://github.com/mlperf/training

14 P. A. Merolla et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science* **345**(6197), 668–673 (2014).

15 https://en.wikipedia.org/wiki/Discrete_event_simulation

16 https://www.spec.org/cpu2006/Docs/471.omnetpp.html

17 https://omnetpp.org/

18 http://iss.ices.utexas.edu/?p=projects/galois/benchmarks/discrete_event_simulation

19 Aamer Jaleel, "Memory Characterization of Workloads Using Instrumentation-Driven Simulation", http://www.jaleels.org/ajaleel/publications/SPECanalysis.pdf (2010)

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

21 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.

22 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.

23 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.

24 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.

25 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 .

26 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, slides: http://extremecomputingtraining.anl.gov/files/2017/08/ATPESC_2017_Dinner_Talk_6_8-4_Kogge-Data_Intensive_Computing.pdf

27 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.

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

29 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.

30 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.

31 754-2008 - IEEE Standard for Floating-Point Arithmetic, IEEE, August 2008, http://ieeexplore.ieee.org/servlet/opac?punumber=4610933

32 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.

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

34 http://www.spec.org/cpu2006/Docs/429.mcf.html

35 http://www.spec.org/cpu2006/Docs/473.astar.html

36 Aamer Jaleel, "Memory Characterization of Workloads Using Instrumentation-Driven Simulation", http://www.jaleels.org/ajaleel/publications/SPECanalysis.pdf (2010)

37 http://spec.org/gwpg/gpc.static/vp12.1info.html