# INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS

2016 EDITION

# APPLICATIONS BENCHMARKING WHITE PAPER

# APPLICATIONS BENCHMARKING

## 1 CHARTER AND MISSION

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

## 2 SCOPE

The scope of AB is all application domains of interest to the users of computing devices across all market drivers.

## 3 CROSS TEAM INTERACTIONS

The AB IFT provides a list of applications drivers (application areas) and their anticipated performance trends to the Systems and Architectures (SA) International Focus Team. The AB IFT receives a list of market drivers from the SA IFT. AB IFT then generates a ***Cross Matrix*** that shows which application areas are important or critical to a given market driver.

## 4 STAKEHOLDERS

Stakeholders for the AB IFT include all users of computing devices. The AB IFT is the voice of these users for the IRDS. All system vendors across all market drivers are also stakeholders in AB IFT as these vendors seek to respond to their customers.

## 5 TECHNOLOGY STATUS, NEW REQUIREMENTS AND POTENTIAL SOLUTIONS

### 5.1 TECHNOLOGY STATUS AND UPDATE

The AB IFT is new to IRDS. Thus this document outlines its status but does not provide an update from the ITRS 2.0 roadmap. The Applications Areas identified by the AB IFT are shown below in Table 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 1 Application Areas**

| Application area | Desired metric | Description |
|---|---|---|
| Big Data Analytics | Feature/sec | Data mining to identify nodes in a large graph that satisfy a given feature/features |
| Feature Recognition | Feature/sec | 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 | Sim/sec | 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 | Sim/sec | Simulation of physical real-world phenomena. Typically finite-element based. Examples include fluid flow, weather prediction, thermo-evolution. Computation is floating-point-based. |
| Optimization | Solution/sec | Integer NP-hard optimization problems |
| Graphics/VR/AR | Frame/sec | Large scale, real-time photorealistic rendering driven by physical world models. Examples include interactive gaming, Augmented Reality, Virtual Reality. |
| Media processing | Frame/sec | Discrete processing, including filtering, compressing, decompressing of streaming media, where the media is unknown (i.e., camera stream based). Includes integrating multiple cameras to feed graphics rendering |
| Cryptographic codec | Codon/sec | Crypting and decrypting of data at the edge of cryptographic science. Includes asymmetric-key encryption, excludes symmetric-key encryption. |

## 5.2 Requirements and Difficult Challenges

1. **Benchmark Availability:** There are several benchmark sets available that cover each application area. However, many of these benchmarks either only cover 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.
3. **Metrics to Track:** The metrics to track listed in Table 1 are not easily derived from all benchmark scores.
4. **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).
5. **Power/Energy Scoring:** With a few exceptions, the majority of available benchmarks track metrics that are disconnected from power dissipation and total energy consumption.

## 5.3 Potential Solutions

1. **Benchmark Availability**: Over time, the plan is for AB IFT will develop new benchmark sets that more appropriately measure the desired application areas identified by AB IFT.

2. **Benchmark Results Availability:** Once AB IFT-designed benchmarks are made available, members of IRDS AB and their employers will control the availability of results.
3. **Metrics to Track:** For today, the AB IFT must use whatever metrics were collected. Once AB-IFT benchmarks are made available, these benchmarks will be designed to capture the desired metrics.
4. **Metrics vs. Precision and Accuracy:** Currently, AB IFT will use benchmark results that are consistent in their accuracy and precision specifications.
5. **Power/Energy Scoring:** AB IFT will use a platform form factor (e.g., mobile phones, server blades, etc.) as a proxy for power/energy efficiency. More desirable, but infeasible at this time, is to use the IEEE RCI Low Power Image Recognition Competition approach (http://rebootingcomputing.ieee.org/lpirc).

## 5.4 APPLICATION AREA: "BIG DATA ANALYTICS"

The Big Data Analytics application area focuses on Data mining to identify nodes in a large graph that satisfy a given feature/features.

**Potential Public Benchmarks**

**Graph 500**

The Graph 500 benchmark (http://www.graph500.org/specifications) contains two kernels accessing a single data structure representing a weighted, undirected graph. The first kernel constructs the graph from the input tuple list; the second one operates on the graph. The first kernel constructs an *undirected* 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 remaining kernels. For instance, kernel 1 may construct a (sparse) graph from a list of tuples; each tuple contains endpoint vertex identifiers for an edge, and a weight that represents data assigned to the edge. There are various internal memory representations for sparse graphs, including (but not limited to) sparse matrices and (multi-level) linked lists.

Kernel 2: Breadth-first Search:

A Breadth-First Search (BFS) of a graph starts with a single source vertex, then, in phases, finds and labels its neighbors, then the neighbors of its neighbors, etc. This is a fundamental method on which many graph algorithms are based. They do 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 (internal or external) is data-dependent with small average prefetch depth. As in a simple concurrent linked-list traversal benchmark, performance reflects an architecture's throughput when executing concurrent threads, each of low memory concurrency and high memory reference density. Unlike such a benchmark, this one also measures resilience to hot-spotting when many of the memory references are to the same location; efficiency when every thread's execution path depends on the asynchronous side-effects of others; and the ability to dynamically load balance unpredictably sized work units. The benchmark has 64 BFS searches from different sources executed in sequence.

Problem Sizes:

| Problem class | Scale | Approx. storage size in TB |
|---|---|---|
| Toy (level 10) | 26 | 0.0172 |
| Mini (level 11) | 29 | 0.1374 |

| | | |
|---|---|---|
| Small (level 12) | 32 | 1.0995 |
| Medium (level 13) | 36 | 17.5922 |
| Large (level 14) | 39 | 140.7375 |
| Huge (level 15) | 42 | 1125.8999 |

Performance Metrics

For Kernel 2, they define a new rate called traversed edges per second (TEPS) as follows. Let $\text{time}_{K2}(n)$ be the measured execution time for kernel 2. Let m be the number of input edge tuples within the component traversed by the search, counting any multiple edges and self-loops. They define the normalized performance rate (number of edge traversals per second) as:

$\text{TEPS}(n) = m / \text{time}_{K2}(n)$

Output results:

- The output must contain the following information (among others):
- SCALE: Graph generation parameter
- Construction_time: The single kernel 1 time
- min_time, firstquartile_time, median_time, thirdquartile_time, max_time: Quartiles for the kernel 2 times
- mean_time, stddev_time: Mean and standard deviation of the kernel 2 times

RESULTS

Figure 1 plots the number of GTEPS of the top three machines in the last
6 years. The data is refreshed every 6 months. Nearly all of these runs are with the "Large" problem size.

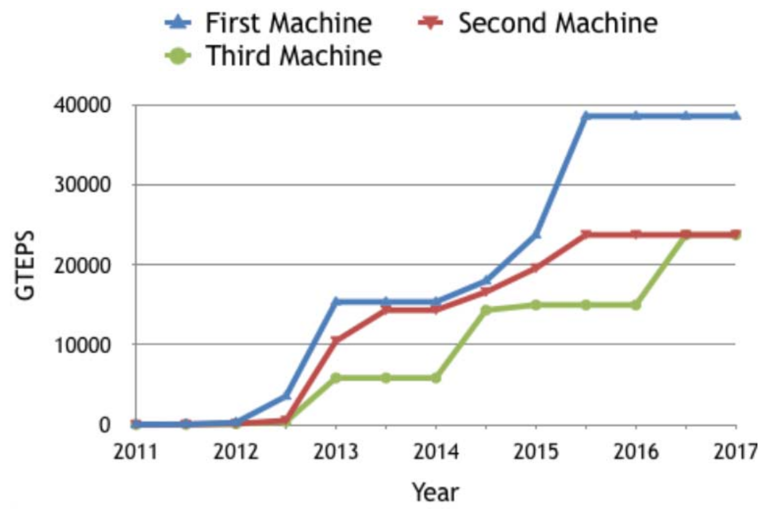
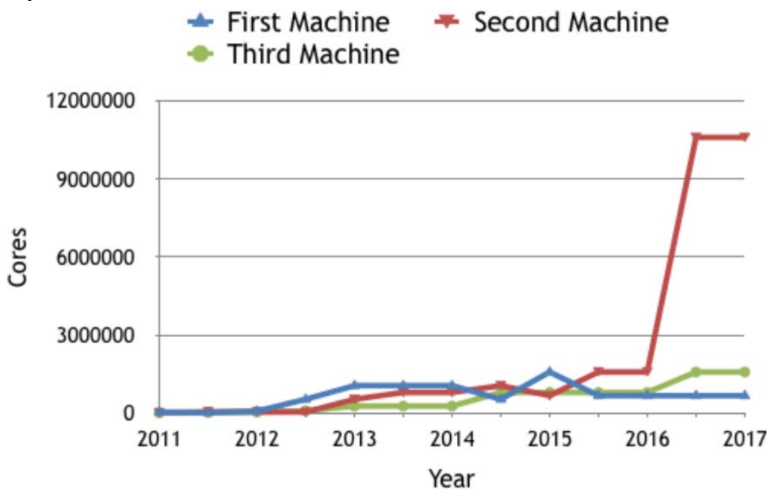


Figure 2 plots the number of cores used in the top three Graph 500 machines in the last 6 years.



**PARSEC**

Parsec (http://parsec.cs.princeton.edu) has a variety of parallel applications. We are interested in two datamining applications:

1. Freqmine: This application employs an array-based version of the FP-growth (Frequent Pattern-growth) method for Frequent Itemset Mining (FIMI). It is an Intel RMS benchmark which was originally developed by Concordia University.
2. Streamcluster: This RMS kernel solves the online clustering problem. For a stream of input points, it finds a predetermined number of medians so that each point is assigned to its nearest center.

There is no execution time available anywhere.

**BigDataBench**

BigDataBench (http://prof.ict.ac.cn/) is a suite of big data applications. There are two applications that use graphs and have good data sets:

1. Google Web Graph [2] — This data set is unstructured, containing 875713 nodes representing web pages and 5105039 edges representing the links between web pages. This data set is released by Google as a part of Google Programming Contest.
2. Facebook Social Network[3] — 4039 nodes, 88234 edges (unstructured graph)

There is also no data on execution time of these applications.

## 5.5 Application Area: "Feature Recognition"

The application area of "Feature Recognition" refers to both the use and the training of Deep Neural Networks (DNNs), as well as any other systems that derive their performance characteristics by learning over large sets of datas. The goal of these benchmarks is to track progress in this field to enable continued evolutionary improvement of existing computational approaches (e.g., CPUs and GPUs), as well as to provide an accurate performance target for more

[2] *Google web graph. http://snap.stanford.edu/data/web-Google.html .*
[3] *Facebook graph. http://snap.stanford.edu/data/egonets-Facebook.html*

revolutionary approaches (approximate computing approaches based either on digital techniques, on analog memory devices, or other approaches).

For this application area, we plan to track two sets of benchmarks:

1. Forward-evaluation of already trained DNNs. Because much of this computation might take place in mobile or other power-constrained platforms, these metrics should include both classifications-per-second and classifications-per-second-per-Watt. Our 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.
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.

Here we list some of the challenges for this application area:

1. 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%. Thus it can be difficult to define the "completion" of training.
2. 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. However, 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, analog techniques.
3. There are numerous ways in which time-per-feature 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 20x <https://github.com/soumith/convnet-benchmarks>. 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.
4. While there are some emerging benchmarks that focus on essential component blocks of DNN systems (see https://svail.github.io/DeepBench/, 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.

**Potential Public Benchmarks**

The approach we propose here is 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.

Since each example is well-known and defined (in terms of number of layers, number of neurons and synapses, the add-ons such as dropout and momentum that were used when the network was first published), we can 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 then provides 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.

For convolutional networks, existing data on time-for-training can be used from < https://github.com/jcjohnson/cnn-benchmarks>. Despite the fact that AlexNet is well-known and comparatively small in size, we choose to avoid this network because of its use of normalization within the network layers. Normalization can consume considerable computational resources, yet is not a component needed or used in later CNN networks. Thus we choose to track VGG-

16 < https://github.com/jcjohnson/cnn-benchmarks#vgg-16, https://github.com/jcjohnson/cnn-benchmarks#vgg-paper, https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md >, showing results for the same cuDNN software framework for an input of size 16 x 3 x 224 x 224 (e.g., 16 examples per minibatch). Note that power information was not included in the posted information. It is not clear if we can estimate power from the power specifications of the underlying hardware, unless we know "how much" of the GPU's capabilities (IO bandwidth, compute capabilities) were needed for the benchmark. It is quite likely that the maximum power is not required unless the network in question used all of the bandwidth AND all of the compute resources of the hardware.

Also, it is not clear whether the forward-evaluation numbers shown here could be improved by pipelining or other techniques that only become feasible for a forward-evaluate-only implementation. (Note also that we will need to include here an accuracy target associated with this network, averaged over several different sets of random starting weights.)

**Table 2 Benchmarking results for the VGG-16 network**

| GPU | cuDNN | Forward (ms) | Backward (ms) | Total (ms) |
|---|---|---|---|---|
| Pascal Titan X | 5.1.05 | 41.59 | 87.03 | 128.62 |
| GTX 1080 | 5.1.05 | 59.37 | 123.42 | 182.79 |
| Maxwell Titan X | 5.1.05 | 62.30 | 130.48 | 192.78 |
| CPU: Dual Xeon E5-2630 v3 | None | 3101.76 | 5393.72 | 8495.48 |

Similar data will be needed for one representative FC-DNN, one representative LSTM, and any other networks that should be considered. For each representative network and its associated DNN dataset (see http://deeplearning.net/datasets/), a target accuracy will need to be defined and power (or total energy) accurately measured. For runs on conventional GPU hardware, the software framework used should be fully identified (as it was in the data used to generate Table 1), to aid in comparing different measurements.

## 5.6 Application Area: "Discrete Event Simulation"

The discrete event simulation application area focuses on systems that are used to simulate other systems, typically via discrete-event methods.

**Potential Public Benchmarks**

1. 471.omnetpp from SPEC CPU2006: https://www.spec.org/cpu2006/Docs/471.omnetpp.html
   a. Description:
      i. The benchmark performs discrete event simulation of a large Ethernet network. The simulation is based on the OMNeT++ discrete event simulation system (www.omnetpp.org), a generic and open simulation framework. 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. The Ethernet model used in this benchmark is publicly available from the address given in the References.
      ii. For the reference workload, the simulated network models a large Ethernet campus backbone, with several smaller LANs of various sizes hanging off each backbone switch. The model contains altogether about 8000 computers (hosts), and 900 switches and hubs. It mixes all kinds of Ethernet technology: Gigabit Ethernet, 100Mb full duplex, 100Mb half duplex, 10Mb UTP, 10Mb bus ("thin Ethernet"), switched hubs, repeating hubs.
      iii. The training workload models a smaller LAN with several hubs and busses.

iv. The model is accurate in the sense that the CSMA/CD protocol of Ethernet and the Ethernet frame are faithfully modelled. The host model contains a traffic generator which implements a generic request-response based protocol. (Higher layer protocols are not modelled in detail.) With appropriate configuration, the traffic generator can be tuned become a rough model for protocols such as SMB/CIFS (the Windows file sharing protocol), HTTP, or a database client-server protocol.

b. Large number of performance simulation should be available as part of SPEC

2. DES benchmark (circuit simulation of netlists such as adders) from Galois suite: http://iss.ices.utexas.edu/?p=projects/galois/benchmarks/discrete_event_simulation
   a. Description
      i. The Discrete Event Simulation (DES) algorithm is commonly used to simulate systems of interacting stations, e.g., circuit simulation, units in a battle field, assembly line etc. The interacting objects in the system are modeled as stations, which store some state. During the course of simulation, the stations interact with each other via messages or events, and cause changes in their own state or the state of other stations. The changes are modeled as events being sent from a station to other stations. An event causes the target station to potentially update its state and generate new events destined for other stations in the system dependent upon this change. The system is represented as a graph where nodes are the stations and edges represent the communication links between the stations. Every event has a time stamp associated with it. The system evolves by processing the events in their time stamp order. The data parallelism in this algorithm arises from multiple events executing on different stations in the system. This benchmark uses Discrete Event Simulation to model logic circuit simulation. Here the processing stations are logic gates, which are interconnected with each other through wires (edges in the graph). The events in the system represent a change of value on the input or output of a gate. The simulation starts with some predefined initial events for the inputs of the circuit. Processing these initial events causes additional events to be generated in the system. When the input of a gate changes, it updates its output and generates new events to update the inputs of the gates in its fanout and so on. The simulation finishes when there are no unprocessed events.
   b. Performance
      i. Some performance available on website.

## 5.7 Application Area: "Physical System Simulation"

The Physical System Simulation application area deals with simulation of physical, real-world phenomena. These applications typically employ finite-element based techniques. Examples include fluid flow, weather prediction, and thermo-evolution. In contrast to the Discrete Event Simulation application area, Physical System Simulation is dominated by floating-point-based computation.

**Potential Public Benchmarks**

Taking the application programmers and the end-users point of view, the following major benchmarking efforts could be considered – most of them with readily available code implementations:

A. The NAS Parallel Benchmarks (NPB) [1] includes the descriptions of several (initially eight) "pencil and paper" algorithms [2]. All of them are realistically kernels although the authors claim that they include three "simulated applications" but this is 25 years ago. The NPB are flat (no hierarchy) and cover one application domain - CFD - of primary interest for 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 supposed to express the performance of higher level codes by the codes in the layer below. This proved to be a difficult task particularly with the compact application layer including various computational science codes.

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 a few popular kernels including LINPACK. The PARKBENCH suite adopted the hierarchical approach from GENESIS together with the difficulties explained above.

D. All major machine vendors have participated in the development of SPEComp2001. Achieving portability across all involved platforms was an important concern in the development process. The goal was to achieve functional portability as well as 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 we accommodated several requests by individual participants to add small code modifications that take advantage of key features of their machines.

E. Another more recent "pencil and paper" parallel benchmark suite is the Dwarfs Mine [http://view.eecs.berkeley.edu/wiki/Dwarf_Mine] based on the initial “Seven Dwarfs” proposal [5]. 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 NAS PB) was then extended to 13 in the "View from Berkeley" TR [6]. The report confirms "presence" of the 13 Dwarfs in 6 broad application domains (pp. 16-19) – Embedded Computing, GP Computing, Machine Learning, Graphics/Games, Databases and Intel's RMS (recognition/mining/synthesis).

The Dwarfs Mine description, which was followed by a reference implementation effort [7], adopts a bottom-up approach similar to GENESIS and then PARKBENCH although much more systematic. Some more 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 6 application domains described in the "View from Berkeley" TR.

The currently proposed benchmark codes that cover the “Physical System Simulation” application area of interest adopts the bottom-up approach and come in a 2-layer hierarchy. At the lower layer, we are planning to use the “Seven Dwarfs” or kernels with some of them mapped to the NAS PB codes. The upper layer will include several compact representative application benchmarks in relevant areas such as climate modeling, plasma physics (fusion), and medical imaging. Repeatability of the benchmark results will be closely monitored with corresponding analysis and interpretation. First of all, we are aiming to select codes that conform to our repeatability requirements but we also recognize that in a number of cases repeatability properties depend heavily on the computer system and environment which requires more detailed consideration by the IRDS AB ITF.

## 5.8 Application Area: “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 (TST). 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.

**Potential Public Benchmarks**

Near-optimal techniques:

1. SPEC2k6: http://www.spec.org/cpu2006/Docs/429.mcf.html although this solves a vehicle scheduling problem and uses the (accurate, non-NP-hard) network simplex algorithm.

2. SPEC2k6: http://www.spec.org/cpu2006/Docs/473.astar.html, 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.

True optimal techniques:

4. TSPLIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Available results from TSPLIB and more are here: http://plato.asu.edu/bench.html

This tests different packages, thus is more of a codeless benchmark across algorithms and software implementations. Not as useful as the SPEC2k6 benchmarks. However, it appears to be the best choice to capture true optimization techniques. Benchmarks must be created from TSPLIB. AB IFT then needs to run these benchmarks on a variety of platforms.

## 5.9 Application Area: "Graphics/VR/AR"

The Graphics / Virtual Reality / Augmented Reality application area focuses on the challenging and market-defining problem of using high-performance graphics to create virtual environments or augment reality.

The Media Processing application area focuses on processing of media streams, including compression, filtering, transformations and modulations. This is also a critical application area that important in markets such as mobile and internet-of-things edge devices.

For both of these application domains, benchmarks exist but are focused on specific markets. We seek to define a small set of benchmarks that capture these two areas with more generality.

## 5.10 Application Area: "Cryptographic Codec"

The cryptographic codec application area is critical to many emerging and existing markets.

There are two sub-areas in this application area: primitives to do hashing or do an encryption scheme, and full system (e.g., TLS).

**Potential Benchmarks**

EEMBC is creating a benchmark for this space. Their approach is to have benchmarks which will change based on platform. Application spaces include authentication, secure data storage, secure data communications. If we choose to adopt this, we need to focus on common elements for IRDS AB.

A key question is, "how do you account for the presence of acceleration in hardware for encryption?" There is no API that's standardized across these platforms. One potential solution is to use the Android API for security for mobile (note however that that doesn't work for IoT) EEMBC has some security experts working on this issue. We will continue to monitor their progress.

Standards organizations in this space include the IoT Security Foundation, IPSO, etc. Also NIST is making hard requirements and even has a conference dedicated to this: https://www.nist.gov/news-events/events/2016/10/lightweight-cryptography-workshop-2016.

There are few talks that directly relate to the issues with this applications domain:

https://www.nist.gov/sites/default/files/documents/2016/10/17/sonmez-turan-presentation-lwc2016.pdf

https://www.nist.gov/sites/default/files/documents/2016/10/17/schaumont-presentation-lwc2016.pdf

Cryptolux is another potential source for benchmarks (https://www.cryptolux.org/index.php/FELICS).

## 6 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 is the preliminary cross matrix that was developed, where "G" means critical (or "gating") application area(s), "X" means important application area(s), and "P" means important *and* power-constrained application area(s). These three classifications remain open to debate in the AB IFT.

| 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 |
| Media processing | X | X | P | G |
| Cryptographic codec | G, P | X | G. P | G, P |

## 7 SUMMARY

The Applications Benchmarking (AB) International Focus Team's mission is to identify key application drivers, track and roadmap the performance of these applications. We identify 11 application areas that span a broad range of computation. Given a list of market drivers from the Systems and Architectures FT, AB generates a cross matrix map showing what application(s) are important or critical (gating) for each market. The current cross matrix has four area: IoT-e, CC CPS and Mo. Together these represent key markets important to the overall electronics industry.

## 8 References

1. 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, http://www.davidhbailey.com/dhbpapers/benijsa.pdf, Int. J. of Supercomputer Applications, vol. 5(3), 63-73, 1991.

2. D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishman and S. Weeratunga, "The NAS Parallel Benchmarks," https://www.nas.nasa.gov/assets/pdf/techreports/1991/rnr-91-002.pdf, NASA TR RNR-91-02, 1991; https://www.nas.nasa.gov/assets/pdf/techreports/1994/rnr-94-007.pdf, updated NASA TR RNR-94-007, 1994.

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, vol. 8 of Advances in Parallel Computing, Elsevier Science Publishers, 257-271, 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," Technical Report: UT-CS-93-213, http://www.davidhbailey.com/dhbpapers/parkbench.pdf, Scientific Programming, vol. 3(2), 101-146, 1994.

5. P. Colella, Defining Software Requirements for Scientific Computing, DARPA HPCS, 2004, http://view.eecs.berkeley.edu/w/index.php?title=Special:Upload&wpDestFile=DARPAHPCS.ppt (login required).

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, http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf, University of California, Berkeley, TR UCB/EECS-2006-183, Dec. 2006.

7. A. Kaiser, S. Williams, K. Madduri, K. Ibrahim, D. Bailey, J. Demmel, E. Strohmaier, TORCH Computational Reference Kernels: A Testbed for Computer Science Research, Lawrence Berkeley National Laboratory, TR LBNL-4172E, 2011. http://escholarship.org/uc/item/8n36z5tn.pdf, Also, EECS Department, University of California at Berkeley, TR UCB/EECS-2010-144, 2010.

8. C. Addison, V. Getov, A. Hey, R. Hockney, I. Wolton. Benchmarking for Distributed Memory Parallel Systems: Gaining Insight, from Numbers. Parallel Computing, vol. 20(11), 1653-1668, 1994.

9. 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 vol. 2104, pp. 1-10, Springer, 2001.

## 9 ACKNOWLEDGMENTS

Membership of the AB IFT:

| Name | Application Area | Representing |
|---|---|---|
| Geoffrey Burr | Feature Recognition | IBM |
| Douglas Carmean | Graphics/VR/AR | Microsoft |
| Tom Conte [chair] | Optimization, Cryptographic Codec | Georgia Tech & IEEE RCI |
| Yoshihiro Hayashi | | Renesas |
| Markus Levy | Cryptographic Codec | Chair of EEMBC |
| Jonathan Gallmeier | Media Processing, Physical System Simulation | AMD |
| Paolo Gargini | | Chair of IRDS |
| Vladimir Getov | Physical System Simulation | University of Westminster, UK |
| Naveen Muralimanohar | Feature Recognition | HPE |
| Dam Sunwoo | Discrete Event Simulation | ARM |
| Josep Torrellas | Big Data Analytics | UIUC |

## APPENDIX: COMMON BENCHMARK SUITES

### SPEC CPU 2006

Source: https://www.spec.org/cpu2006/publications/CPU2006benchmarks.pdf

Integer:

- 400.perlbench: (programming language)
- 401.bzip2: (Compression)
- 403.gcc: (C Language optimizing compiler)
- 429.mcf: Combinatorial optimization / Single-depot vehicle scheduling
- 445.gobmk: Artificial intelligence - game playing
- 456.hmmer: Search a gene sequence database
- 458.sjeng: Artificial Intelligence (game tree search & pattern recognition)
- 462.libquantum: Physics / Quantum Computing
- 464.h264ref: Video compression
- 471.omnetpp: Discrete Event Simulation
- 473.astar: Computer games, Artificial Intelligence. Path finding
- 483.xalancbmk: XSLT processor for transforming XML documents into HTML, text, or other XML document types

Floating point

- 410.bwaves: Computation Fluid Dynamics
- 416.gamess: Quantum chemical computations
- 433.milc: Physics / Quantum Chromodynamics
- 434.zeusmp: Physics / Magnetohydrodynamics
- 435.gromacs: Chemistry / Molecular Dynamics
- 436.cactusADM: Physics / General Relativity
- 437.leslie3d: Computational Fluid Dynamics (CFD)
- 444.namd: Scientific, Structural biology, classical molecular dynamics simulation
- 447.dealII: Solution of Partial Differential Equations using the Adaptive Finite Element Method
- 450.soplex: Simplex Linear Program (LP) Solver
- 453.povray: Computer visualization (ray-tracing)
- 454.calculix: Structural Mechanics
- 459.GemsFDTD: Computational Electromagnetics (CEM)
- 465.tonto: Quantum Crystallography
- 470.lbm: Computational Fluid Dynamics, Lattice Boltzmann Method
- 481.wrf: Weather Forecasting
- 482.sphinx3: Speech Recognition

## PARSEC

Source: http://parsec.cs.princeton.edu/doc/parsec-report.pdf

- blackscholes: (Financial Analysis) calculates the prices for a portfolio of European options analytically with the Black-Scholes partial differential equation (PDE)
- bodytrack: (Computer Vision) tracks a 3D pose of a marker-less human body with multiple cameras through an image sequence.
- canneal: (Engineering) cache-aware simulated annealing (SA) to minimize the routing cost of a chip design
- dedup: (Enterprise Storage) compresses a data stream with a combination of global compression and local compression in order to achieve high compression ratios. (deduplication)
- facesim: (Animation) it takes a model of a human face and a time sequence of muscle activations and computes a visually realistic animation of the modeled face by simulating the underlying physics.
- ferret: (Similarity Search) content-based similarity search of feature-rich data such as audio, images, video, 3D shapes and so on.
- fluidanimate: (Animation) extension of the Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid for interactive animation purposes
- freqmine: (Data Mining) array-based version of the FP-growth (Frequent Pattern-growth) method for Frequent Itemset Mining (FIMI)
- streamcluster: (Data Mining) solve the online clustering problem. For a stream of input points, it finds a predetermined number of medians so that each point is assigned to its nearest center.
- swaptions: (Financial Analysis) uses the Heath-Jarrow-Morton (HJM) framework to price of portfolio of swaptions.
- vips: (Media Processing) based on the VASARI Image Processing System (VIPS). Includes fundamental image operations such as an affine transformation and a convolution.
- x264: (Media Processing) H.264/AVC (Advanced Video Coding) video encoder.