White Paper | SPECviewperf 12 & BENCHMARKING

TABLE OF CONTENTS

INTRODUCTION

SPECviewperf® 12 is the latest version of the SPECviewperf benchmark released by the Standard Performance Evaluation Committee's (SPEC) Graphics Performance Characterization (SPECgpc) working group. It replaces SPECviewperf® 11, which was released in June 2010. SPECviewperf 12 includes updated versions of SPECviewperf 11 tests as well as new tests to simulate energy and medical applications. SPECviewperf 12 also includes the first DirectX® test from the SPECgpc group.

DESCRIPTION

The main features of the SPECviewperf benchmark is that it is license free, it does not require any third party software licenses to run, and that it uses graphics traces -- the list of graphics commands generated from real applications in an attempt to simulate graphics workloads that users of those applications might encounter. This methodology has both advantages and disadvantages. On the positive side, anyone can run SPECviewperf. As it does not require licenses, anyone can download the benchmark from SPEC and run the test (as long as the test system meets the minimum requirements for the benchmark), and compare results for various workstation configurations. SPEC does have restrictions as to use of the data for non-members, details are specified on the SPEC website, at www.spec.org. A downside to this approach is that it is not possible for SPECviewperf to completely simulate an actual application.

SYNTHETIC BENCHMARKS

Synthetic benchmarks are benchmarks designed to mimic a particular workload. Application benchmarks would use the actual application, resulting in a more accurate representation of the workload. SPECviewperf is a synthetic benchmark. The graphics traces it uses are taken from an application and then used by SPECviewperf to replicate the workload (except for the energy and medical viewset tests, these are purely synthetic and will be described in detail later). The trace consists of the graphics commands used to generate a single graphics frame for a specific data set using a specific set of options specified by the application. There are pros and cons to this approach. On the positive side, using graphics traces from actual applications produces a test that features the graphics features actually used by that application. The downside to this approach is that SPECviewperf traces only capture the graphics calls made by the application, they do not attempt to replicate the applications actual rendering logic. Consider the following sample application pseudo code:

```
Draw some graphics ( );
Sleep for 1 day;
Draw some more graphics stuff ( );
```

In the example above, if the graphics rendering for each of the "Draw some graphics ()" functions take 10 seconds to complete, 20 seconds total for both of them to execute. It would take one day and 20 seconds for all of this code to execute. If SPECviewperf were to extract a trace of the graphics call made from the example above, this trace would look like the following:

```
Draw some graphics ( );
Draw some more graphics stuff ( );
```

So if this graphics trace were implemented in SPECviewperf, as the graphics portions take 20 seconds to complete, the test would take about 20 seconds to complete (the original 20 seconds plus any overhead added by SPECviewperf itself). In this example, the graphics drawn from the SPECviewperf test would look the same as the application, but it would not accurately represent what an end-user would see with the actual application: the application takes more than a day to complete, the SPEC test takes 20 seconds. This is an extreme example, but it highlights the deficiencies of using only a graphics trace from an application to replicate what it does. Using this technique, it's an approximation of the workload from the actual application that is generated; it depends on the actual application as to how well it replicates the application's actual performance.

The following example highlights an additional issue with the graphics trace method. Consider the following pseudo code from a hypothetical application:

```
Until the user quits the application do {

  Draw some graphics and display a frame ( );
  Do some computational stuff that takes
  about 10 minutes ( );

  } end loop
```

If SPECviewperf were to capture the graphics trace from this application, the pseudo code for SPECviewperf would look like this:

```
Until the benchmark test is finished do {

  Rotate the model around before drawing
  next frame ( );
  Draw some graphics and display a frame ( );

  } end loop
```

As you can see from comparing the two samples above, there are significant differences. In the original application code, some computational operations occur between each rendered frame of graphics. As SPECviewperf only uses the graphics traces, this work is not represented in the benchmark. Another difference in the second sample, in the SPECviewperf code, you will notice that the model in the graphics frame is rotated before each frame is rendered. SPECviewperf tests manipulate the models for each frame, usually rotating them, to replicate some kind of graphics workload. This would be similar to a user moving the mouse around to arbitrarily rotate a model in an actual application. As it is highly unlikely many users sit in front of the computer wiggling the mouse around to arbitrarily move a model around very often, the workloads simulated by SPECviewperf are fairly uncommon. In order to create a benchmark test, SPECviewperf must insert additional graphics commands into the original graphics trace to manipulate the models during the test. These additional commands are not part of the original application, adding to the synthetic nature of SPECviewperf.

Even though SPECviewperf is synthetic, it can be used along with other tools to help determine graphics performance. Using other tests, such as application benchmarks and end user testing can help create a more complete picture of graphics performance.

BENCHMARKING GRAPHICS PERFORMANCE

Using benchmarks to determine graphics performance is a complex task. It requires the user to understand the benchmark, its design, what it measures, how it measures, and to be able to interpret the results. Benchmarks only measure what they are designed to measure and these measurements may be valid for only a specific period of time. Benchmarks are like any other piece of software, there may be good benchmarks and some that are not very good. In some cases, it's difficult for an end user to know which benchmarks are most relevant. The key to understanding benchmarks and their relevance is to understand the capabilities of the target hardware and to run multiple tests and do a detailed interpretation of the results. An example of comparing benchmarks will illustrate some techniques that can be used to more fully understand these techniques.

For our discussion, we will use several benchmarks to compare two graphics cards, the AMD FirePro™ W5000 and the Nvidia Quadro K2000, designated by both companies as a mid-range workstation graphics card.

Features	AMD FirePro™ W5000[i]	Nvidia Quadro K2000[ii]
Memory Size	2GB GDDR5	2GB GDDR5
Memory Interface	256-bit	128-bit
Memory Bandwidth	102.4 GBps	64 GBps
Polygons / sec	1.65M	1.3M

From the raw specifications, it would appear that the AMD FirePro W5000 would be the faster card as it has a faster memory interface, more memory bandwidth and can draw more polygons per second than the K2000. A proper investigation would require we run some benchmarks to test this. For our first test, we will run the SPECviewperf 11 test. If we compare the performance of the maya-03 test (test with traces taken from Autodesk® Maya® application), we see the follow results:

SPECviewperf 11	AMD FirePro™ W5000	Nvidia Quadro K2000
maya-03[iii]	72.17	81.26

The results of this benchmark would run counter to the raw performance data we reviewed before. If we were to use this single data point, one could conclude that the K2000 demonstrates better performance with the Autodesk Maya application. SPEC also produces an application benchmark that uses the actual Maya application, so we have an additional test that we can run. The results from this test for these cards are as follows:

SPECapc® for Autodesk Maya 2012	AMD FirePro™ W5000	Nvidia Quadro K2000
Graphics composite[iv]	3.17	2.74

The results from this benchmark are just the opposite, with the AMD FirePro W5000 coming out with the better score. We can take a look at a third test to help determine the better performing card. If we look at the SPECviewperf 12 maya-04 test, we see the following results:

SPECviewperf 12	AMD FirePro™ W5000	Nvidia Quadro K2000
Maya-04[v]	33.08	20.47

Again in this test case, the AMD FirePro W5000 demonstrates higher performance than the K2000 in line with the assumptions drawn from comparing the specifications of the two cards. Why did SPECviewperf 11 show a discrepancy? Reviewers will need to run additional tests and analyze the data in detail to understand this. We can look at another of the SPECviewperf 11 test scores, the lightwave-01 test created from traces from the NewTek™ LightWave application:

SPECviewperf 11	AMD FirePro™ W5000	Nvidia Quadro K2000
Lightwave-01 [vi]	73.89	80.47

But if we compare this score with the SPECapc LightWave 9.6 benchmark that uses the actual application to measure performance we get the following results:

SPECapc LightWave 9.6	AMD FirePro™ W5000	Nvidia Quadro K2000
Interactive[vii]	3.55	3.45

So a closer examination of SPECviewperf 11 scores to the corresponding application benchmarks show significant discrepancies. If you were to rely on SPECviewperf 11 tests alone to determine graphics card performance you would end up drawing the wrong conclusions. This small example with one set of graphics cards highlights the importance of using more than a single test to determine graphics card performance and carefully analyzing the results to make sure that the conclusions drawn from the tests are correct.

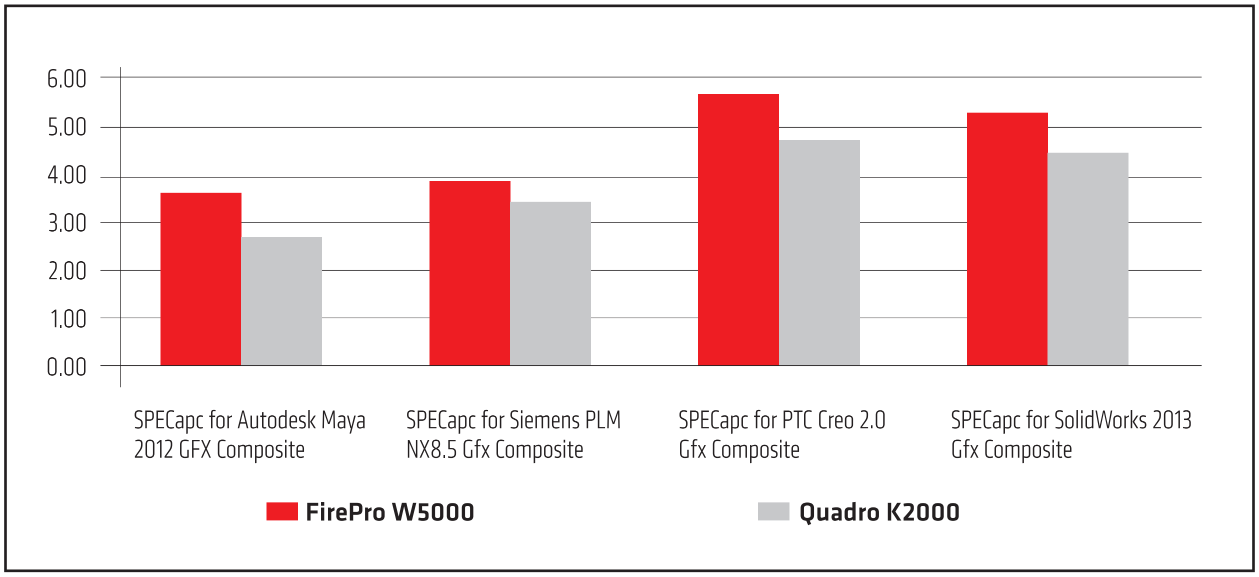
SPECVIEWPERF 12

We have seen that SPECviewperf 11 results did not correlate well with expected raw graphics card performance or benchmarks testing the actual applications traced by SPECviewperf. Is SPECviewperf 12 any better? It is substantially different than its predecessor. After SPECviewperf 11, the benchmark was re-architected to decouple the actual tests and data from the test framework. This design provides for several benefits. First, it allows for viewsets (the individual tests) to be submitted individually. SPECviewperf 11 was created by a single member of the SPEC committee with little ability for other committee members to contribute or review the benchmark dataset source or raw trace data. SPECviewperf 12 consists of viewsets submitted from several committee members, each submitting full source code for the tests that can be easily reviewed by all committee members. Additionally, SPECviewperf 12 traces are taken from the latest versions of the applications. SPECviewperf 11 was released in 2010, with software vendors commonly releasing update versions on an annual basis. SPECviewperf 11 traces are three or more versions behind the currently shipping versions of those applications. SPECviewperf 12 also includes the first DirectX test with traces taken from the Autodesk Showcase® application. There are also tests designed to emulate workloads for energy and medical volumetric viewing applications. These viewsets are entirely synthetic; they were not traced from any specific application, being submitted as original source code by NVIDIA.

If we compare the two graphics cards discussed earlier, the AMD FirePro™ W5000 and the NVIDIA Quadro K2000, looking at the results from SPECviewperf 12 and any corresponding SPECapc application benchmarks, we can see if SPECviewperf 12 is doing a better job with respect to its scores corresponding to the application tests than SPECviewperf 11. Here are the SPECviewperf 12 test scores for each card[viii]:

SPEC Viewperf 12	AMD FirePro W5000	Nvidia Quadro K2000
catia-04	36.52	20.49
creo-01	27.78	21.41
energy-01	0.50	0.41
maya-04	35.65	20.46
medical-01	11.23	6.42
showcase-01	23.54	12.92
snx-02	46.67	20.38
sw-03	50.71	34.81

In SPECviewperf 12, the AMD FirePro™ W5000 outperforms the NVIDIA Quadro K2000 for all subtests. In order to check this against application level benchmarks, SPEC currently has application benchmarks for Autodesk Maya, Siemens PLM NX, PTC Creo, and Dassault Systemes SolidWorks. We can use these tests to see how well SPECviewperf 12 scores correlate to the results from application level benchmarks. The results of the graphics composite scores for these benchmarks are listed in the chart below[ix]:



As we see in the chart, the AMD FirePro ™ W5000 outperforms the Quadro K2000 for all of the SPECapc application benchmarks as well. SPECviewperf 12 is doing a much better job at indicating how a graphics card will perform with actual applications than SPECviewperf 11.

CONCLUSION

SPECviewperf12 is a welcomed update to the SPECviewperf benchmark. SPECviewperf 11 is showing its age in many ways: scores do not correlate well with actual application performance; it does not exhibit good scaling between GPUs of varying performance levels; and the traces used are 3 to 4 years old or older, so they do not represent the latest versions of the applications very well. SPECviewperf 12 with its newer application traces and heavier testing of raw GPU performance provides users with a much better performance testing tool. No single benchmark can provide all the answers with respect to GPU performance. Using synthetic benchmarks like SPECviewperf 12, in combination with application benchmarks and end-user testing, can help provide a more complete picture as to the actual GPU performance and end user will realize in their environment. Benchmark testing requires a thorough understanding of the hardware and benchmarks being tested, running multiple benchmarks over multiple iterations and configurations, and careful analysis of the resulting data. SPECviewperf 12, in combination with application and end user benchmarks, can be a valuable part of the benchmarking toolkit for those investigating GPU performance.

AMD, the AMD Arrow logo, and FirePro are trademarks of Advanced Micro Devices, Inc. SPEC and SPECviewperf are trademarks of Standard Performance Evaluation Corporation. DirectX is a registered trademark of Microsoft Corporation in the U.S. and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.

i Source: http://www.amd.com/US/PRODUCTS/WORKSTATION/GRAPHICS/ati-firepro-3d/W5000/Pages/w5000.aspx

ii Source: http://www.nvidia.com/object/quadro-desktop-gpus.html

iii SPECviewperf 11; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 9.003.3 driver for W5000, Nvidia 311.35 driver for K2000

iv SPECapc for Autodesk Maya 2012 SP1 benchmark; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 9.003.3 driver for W5000, Nvidia 311.35 driver for K2000

v SPECviewperf 12; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 13.152.4 driver for W5000, Nvidia 331.82 driver for K2000

vi SPECviewperf 11; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 9.003.3 driver for W5000, Nvidia 311.35 driver for K2000

vii SPECapc for NewTek LightWave 9.6; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 9.003.3 driver for W5000, Nvidia 311.35 driver for K2000

viii SPECviewperf 12; test configuration: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, AMD 13.25.18.1 driver for W5000, Nvidia 331.82 driver for K2000

ix Test system: Intel Xeon E-1660@ 3.3GHz, Windows 7 Professional 64-bit SP1, Nvidia driver 331.65 for K2000, AMD driver 12.152.4 for W5000;