PVRTune

Counter List and Description

Filename	:	PVRTune.Counter List and Description
Version	:	PowerVR SDK REL_17.2@4910709a External Issue
Issue Date	:	30 Oct 2017
Author	:	Imagination Technologies Limited

Contents

1. Introduction

1.1. Document Overview

This document provides a full list of default PVRTune counters and their description. The document is applicable to PVRTune application as well as PVRScope.

1.2. Software Overview

PVRTune is a performance analysis tool for the PowerVR graphics cores. It uses driver level counters and hardware registers to provide real-time data on the performance of an application running on a PowerVR graphics core. PVRTune receives performance data from either PVRPerfServer or PVRHub running on the target device.

PVRScope is a performance analysis library that retrieves graphics core counter data and works alongside PVRTune to augment it with custom markers and counters.

2. Counter List

This section contains the detailed explanation of the default PVRTune counters as well as information on how to interpret counter data and values.

Note: There is no single "GPU utilisation" counter; the various cores in the HW are independent enough that a single value would have little meaning. In a typical 3D application we would consider the two counters "Tiler active" and "Renderer active" to give the utilisation of the two major GPU parts - some applications might also have significant "2D active" or "Compute active" percentages.

2.1. 2D Active

What does this counter show?

If the GPU in the target device contains a PowerVR 2D graphics core, this counter will show the load of this unit. If a PowerVR 2D core is not present in the target device the counter will be greyed out.

The purpose of the 2D core is to perform efficient blitting operations. For example, OS composition may utilize the 2D core so that the rest of the GPU pipeline can be dedicated to application rendering.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of 2D task timing data.

What does a high value mean?

A high value indicates that the 2D core has been active for a significant percentage of time. If you believe this is a bottleneck, reducing the number of operations utilising the 2D core (OS surface composition, texture uploads etc.) may improve performance.

2.2. 2D Time Per Frame

What does this counter show?

If the GPU in the target device contains a PowerVR 2D graphics core, this counter shows the time spent processing 2D tasks (in seconds) during the specified period. If a PowerVR 2D core is not present in the target device the counter will be greyed out.

The purpose of the 2D core is to perform efficient blitting operations. For example, OS composition may utilize the 2D core so that the rest of the GPU pipeline can be dedicated to application rendering.

2.3. Active Slot Occupancy

What does this counter show?

This counter represents the occupancy of the shader engine's active task slots. Tasks assigned to active slots can be scheduled in for execution immediately. When a task in an active slot encounters a data dependency, it is de-scheduled and another task takes its place.

What does a high value mean?

A high percentage indicates that the shader engine processing pipelines are well utilized. A low percentage may indicate:

- High data dependency: If a large number of tasks in the available slots are resolving data dependencies, there may not be a sufficient number of ready tasks to schedule into the active slots.
- Low available slot occupancy: If available slot occupancy is low, there may be an insufficient number of ready tasks to populate the active slots.

2.4. Available Slot Occupancy

What does this counter show?

This counter represents the occupancy of the shader engine's available slots. Tasks in these slots may be ready for execution or may be resolving data dependencies. When an active slot becomes vacant, the shader engine will populate it with a ready task from the available slot list.

What does a high value mean?

A high percentage indicates that there are a sufficient number of tasks in the pipeline to feed the active slots. A low percentage may indicate:

- Register pressure: Each occupant task utilises register space for its inputs. If a large amount of registers are required for the occupant tasks (i.e. register pressure is high), there may not be enough space available to schedule tasks into vacant slots.
- ISP bottlenecked: If the pipeline is bottlenecked by the ISP, there may be an insufficient number of tasks sent to the shader engine to populate all available slots.

2.5. Compute Active

What does this counter show?

This counter shows the percentage of time that Compute tasks were active.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of Compute task timing data.

What does a high value mean?

A high value indicates that Compute tasks have been active for a significant percentage of time. If you believe this is a bottleneck, you should reduce the complexity of your compute kernels and/or reduce the number of kernels executed.

2.6. Compute Time Per Frame

What does this counter show?

This counter shows the time spent processing Compute tasks (in seconds) during the specified period.

2.7. CPU Load

What does this counter show?

This counter shows the combined load of all CPU cores.

What does a high value mean?

If the CPU load is very high and there are large gaps between the Tiler and Renderer timing blocks, then the system is most likely CPU limited. If this is the case, the load of the Tiler, Texturing and Shader Processor units will also be low (e.g. 50%).

In a system where there are multiple CPU cores, a value of 100% would indicate that all CPU cores are 100% busy. On a platform with a dual-core CPU, a value of 50% could indicate that one CPU core is 100% busy and the other is idle (0% busy) or it could be caused by the cores being at any other combination of loads that result in a 50% average. If, for example, a single-threaded application is running on a dual-core CPU and the load is 50%, the application might be maxing out one of the cores.

If the value of this counter is very high, you should do the following:

- Run a CPU profiler: Investigate the issue further with a dedicated CPU profiler.
- Enable API profiling: PVRTune can capture timing data for API calls. For more information, please refer to the PVRTune documentation.

2.8. CPU Load Kernel

What does this counter show?

This counter shows the kernel-mode load of all CPU cores.

What does a high value mean?

A high value indicates that kernel-mode processes are causing a CPU bottleneck.

2.9. CPU Load User

What does this counter show?

This counter shows the user-mode load of all CPU cores.

What does a high value mean?

A high value indicates that user-mode processes are causing a CPU bottleneck.

2.10. Cycles Per Pixel

What does this counter show?

This counter represents the average number of cycles that the Shader Processor has spent processing fragments.

It is worth noting that the number of fragments processed by the Shader Processor for a given frame will match the number of fragments that have been submitted for shading by the Texturing and Shading Processor, rather than this value representing the average per screen-pixel. Consider the following scenario: If the only render submitted to the GPU was a single screen aligned full screen opaque quad with a fragment shader that took 20 cycles, then the average value of Cycles per pixel would be 20. If the same opaque quad was rendered with an additional blended screen aligned full screen quad in front of it that took 10 cycles, then the average value of Cycles per pixel would be 15.

What does a high value mean?

A high value indicates that the average cost of shading a fragment is high. As previously mentioned, the value of this counter is easy to misinterpret. It is best to rely on this counter in benchmarks where a single fragment shader is applied to a single full screen opaque quad to test the performance of the shader on the target device.

If this value is high, then you should profile using the PVRTrace's Shader Analysis to isolate the most expensive shaders in your scene and use the built-in shader profiler to optimise them. Alternatively, you could batch process your application's vertex shaders with the appropriate offline profiling compiler in the PowerVR SDK using the “-perfsim” flag to identify the most expensive one in your scene, which will help you find the best place where to focus your optimization.

2.11. Cycles Per Vertex

What does this counter show?

This counter represents the average number of clock cycles that the Shader Engine has spent processing vertices.

What does a high value mean?

A high value indicates that the average cost of shading a vertex is high.

If this value is high, then you should profile using PVRTrace's Shader Analysis to isolate the most expensive shaders in your scene and use the built-in shader profiler to optimize them. Alternatively, you could batch process your application's vertex shaders with the appropriate offline profiling compiler in the PowerVR SDK using the “-perfsim” flag to identify the most expensive one in your scene, which will help you find the best place where to focus your optimization.

2.12. FBA Load

What does this counter show?

This counter represents the average load of the GPU's Frame Buffer Accumulate (FBA) unit. This unit is responsible for a faster type of image access similar to that of image atomics but which does not return results, therefore the execution can be queued up and executed asynchronously.</p>

Accumulation image operations read from a selected texel, compute a new value as described by the function, and write the new value to the selected texel. The contents of the texel being updated by the accumulate function are guaranteed not to be updated by any other accumulate function between the time the original value is read, and the time the new value is written. Image load and store operations perform relatively simple exclusive read or write operations on a given texel, respectively. Image accumulation functions perform comparatively more complex operations and accelerate read/modify/write functions acting atomically with regard to any other accumulation function. It is important to realise that accumulation functions perform only associative operations, such as addition, so the order of their execution is not important.

It should be noted that neither frame shaders nor ray shaders have access to generic image load or store operations, and instead have access only to image accumulation operations using the FBA unit.

What does a high value mean?

A high load value indicates that a very large number of frame buffer accumulation operations have been issued to the FBA.

If this value is very high then the first task carried out should be to identify where and when accumulation operations are issued and determine whether there are cases where accumulations can be combined, or the number reduced altogether. When only necessary accumulations remain you should try the following to reduce the FBA load:

- Avoid any frame buffer accumulations when the result of any such accumulation would have a negligible effect on the resulting image. This is done by checking the potential accumulation value prior to any such accumulation, and skipping the accumulation if the value is below a predefined threshold value. It should be noted that this technique can also be used to reduce the total number of rays issued by checking the data carried by any ray against a threshold value, and skipping the ray emission if the result would be negligible.
- Common usage and general expectations are that for each ray emission there will be one or very few frame buffer accumulations, and therefore usage should be tailored towards these recommendations as far as possible.
- Although it is possible to issue frame buffer accumulates from frame shaders, often it can be more sensible to instead use traditional frame buffer writes from fragment shaders executed at an earlier stage. This will typically execute faster than the comparatively slow frame buffer accumulates. This is more common for hybrid ray traced applications.
- In some cases an application may bounce rays multiple times per frame shader iteration with frame buffer accumulates issued at each ray bounce. However it may be possible instead to defer these accumulations until any such ray terminates i.e. carry the data to be accumulated until a later point/bounce, and only carry out a single accumulate or a reduced number of combined accumulations at the end of the ray chain.

2.13. Frames Per Second (FPS)

What does this counter show?

This counter represents the average number of frames-per-second processed by the GPU over the selected period of time. This value is the reciprocal of the frame time (in seconds).

Similar to the frame time counter, this value is not calculated per CPU process. This counter represents the number of frames the GPU has been able to render within the selected period of time, regardless of which process submitted the rendering task.

On most platforms, V-sync will prevent the number of frames-per-second exceeding the refresh rate of the display.

What does a high value mean?

A high value indicates that the GPU has been able to process a high number of frames during the selected period of time, offering a smooth experience to the user.

2.14. Frame Time

What does this counter show?

This counter represents the average time it has taken the GPU to process a frame (in seconds) over the selected period.

It's worth noting that this value is not calculated per CPU process, i.e. if there are two processes using the GPU to render, the average frame-time over the course of a second will be the average frame-time of both processes during that period of time.

What does a high value mean?

The larger this value is, the longer it takes on average for the GPU to render a frame. It's worth noting that this average includes all processes utilising the GPU during the selected time period so if a second process is running alongside the application you want to profile, e.g. OS composition, then this will affect the average frame time value.

2.15. GPU Clock Speed

What does this counter show?

This counter represents the current clock speed of the GPU. On many modern devices, the GPU clock speed will be change dynamically depending on the workload of the GPU and the thermal limits of the chip.

What does a high value mean?

A consistently low value while the GPU is busy may indicate that the device's internal temperature has exceeded the vendor defined limit. In this scenario, clock speed is reduced in order to reduce thermal load.

2.16. GPU Memory Read Rate (Words Per Second)

What does this counter show?

This counter shows the rate within the current period that the GPU is reading data into memory in Words/sec. To translate Words to bytes requires knowing the word size; this depends on the HW in use, specifically the width of the memory bus (often 16 or 32 bytes).

What does a high value mean?

A constantly high value means the GPU may be accessing memory too often. On shared memory systems it is possible that this value may be low, but the whole system might be still memory bandwidth limited.

If this value is high, then you should reduce number of memory reads. This can achieved by removing texture fetches from the shaders or making textures more cache efficient using texture compression. There can be a small overhead reading vertex data from the Parameter Buffer. In these cases try to reduce the number of polygons or the number of varyings used in the shaders.

2.17. GPU Memory Rate (Words Per Second)

What does this counter show?

This counter shows the rate within the current period that the GPU is reading or writing data in memory in Words/sec. To translate Words to bytes requires knowing the word size; this depends on the HW in use, specifically the width of the memory bus (often 16 or 32 bytes).

What does a high value mean?

A constantly high value means the GPU may be accessing memory too often. On shared memory systems it is possible that this value may be low, but the whole system might be still memory bandwidth limited.

If this value is high, then you should reduce number of memory accesses. See the read and write specific versions of this counter.

2.18. GPU Memory Write (Words Per Second)

What does this counter show?

This counter shows the rate within the current period that the GPU is writing data into memory in Words/sec. To translate Words to bytes requires knowing the word size; this depends on the HW in use, specifically the width of the memory bus (often 16 or 32 bytes).

What does a high value mean?

A constantly high value means the GPU may be accessing memory too often. On shared memory systems it is possible that this value may be a consistently low, but the system is still memory bandwidth limited.

Memory write should be rarely a bottleneck. In the case that this counter is very high, try to reduce writing operations, for example reducing off-screen frame buffers, etc.

2.19. HSR Efficiency

What does this counter show?

This counter represents the effectiveness of the Hidden Surface Removal (HSR) engine, rejecting obscured pixels before they get processed. This tells you the percentage of pixels sent to be shaded, out of the total number of pixels submitted. Any pixel occluded by an opaque polygon and not visible is rejected at this early stage. This avoids the expensive processing and texturing of pixels that are not visible, maximizing processing performance and saving memory bandwidth.

What does a high value mean?

A value of 0% indicates that the number of input pixels in the HSR is the same as the number of pixels to be shaded, implying that no pixels have been rejected. The higher the value, the better rejection rate, achieving the associated performance and bandwidth benefits. For example, a 50% value means that half of the pixels have been rejected before being sent for processing. If this value is very low you should try to reduce the number of blended fragments submitted. It is also recommended to submit opaque draws before translucent ones, as this allows the HSR engine to reject occluded blended fragments. Bear in mind that a simple scene with very little overdraw might show a very small HSR Efficiency value as most pixels are visible.

A 100% value or closer is only possible when most of the pixels have been rejected by the depth comparison which should point to an application problem.

2.20. ISP Overload

What does this counter show?

The Image Synthesis Processor (ISP) is the part of the graphic core that fetches the primitive data and performs Hidden Surface Removal (HSR), along with depth and stencil tests. This counter indicates if this unit has become a bottleneck.

What does a high value mean?

An ISP overload event is quite rare and it occurs when one or several tiles have to process a large amount of overlapped polygons (i.e. high overdraw). To avoid this situation, reduce the amount of geometry submitted to the GPU by performing visibility tests and culling on the CPU.

2.21. ISP Pixel Load

What does this counter show?

Percentage of the time that the ISP pixel-processing is busy.

What does a high value mean?

It would mean that a large quantity of non-visible (non-shaded) pixels are being processed. These may be hidden due to:

- The depth/stencil test (e.g. hidden behind an opaque object).
- A process may be rendering triangles that update only the depth/stencil buffer (and not any colour buffer).

2.22. Leaf Voxels Per Triangle

What does this counter show?

This counter represents the average number of leaf voxels produced per triangle by the Scene Hierarchy Generator Back-end(SHG).

What does a high value mean?

A high value indicates that the generated spatial index structure contains a large number of leaf voxels per triangle.

2.23. Output Pixels Per Frame

What does this counter show?

This counter shows the number of pixels written to system memory per-frame. This number should be identical to the render buffer size in pixels.

What does a high value mean?

A high value means that the number of pixels that need to be rasterized is high, which might put extra pressure on the pixel and texturing units. Features like MSAA could also have an impact at very high resolutions.

2.24. Output Pixels Per Second

What does this counter show?

This counter shows the number of pixels written to system memory per-second.

What does a high value mean?

A high value means that the number of pixels that need to be rasterized is high, which might put extra pressure on the pixel processing and texturing units. Features like MSAA could also have an impact at very high resolutions.

2.25. Overdraw

What does this counter show?

This counter gives the pixel overdraw of the render. 1x overdraw means every on-screen pixel was shaded once.

What does a high value mean?

There is more overdraw. A value of 2 could mean that there is a full-screen opaque object covered by a full-screen translucent object.

2.26. Processing Load Ray

What does this counter show?

This counter represents the average ray workload of the Shader Processor. This average is calculated across the currently selected time range, regardless of whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where Ray Tracing Unit (RTU) tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent carrying out ray shading.

If this value is high, then you should do the following:

- An attempt should be made to identify the highest priority shaders in the scene i.e. those which are both expensive to run and those that are being run frequently. The most expensive ray shader in the scene may very well not be the ray shader taking up the most Shader Processor time as it may not be executed very frequently. The highest priority ray shader can very often be a default ray shader used in a high proportion of ray emissions. It should be noted that any changes to the scene, or to the ray tracing techniques used may result in dramatic changes with regard to which shaders are executing most frequently. Therefore an attempt should be made to optimise all shaders to some extent.
- The ray shaders should then be profiled using the offline compiler.
- The number of ray emissions can also be reduced to reduce the frequency of ray shader executions.

2.27. Processing Load Ray-Vertex

What does this counter show?

This counter represents the average ray-vertex workload of the Shader Processor. This average is calculated across the currently selected time range, regardless of whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where SHG tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent carrying out ray-vertex shading i.e. running vertex shaders used in the construction of a Scene Hierarchy.

If this value is high, then you should do the following:

- An attempt should be made to identify the highest priority ray vertex shaders in the scene i.e. those that are most expensive and run most frequently in the scene.
- The highest priority ray vertex shaders should then be profiled using the offline compiler.
- Improve CPU object culling such as Frustum culling: Improving CPU culling of components so there are fewer components being built into a scene hierarchy will reduce the Shader's ray-vertex processing load, as fewer vertices will be submitted to the GPU. Object culling should be applied only to Component's which will not have any effect on the scene either directly or indirectly.
- Increase vertex sharing: Use index lists and try to have a low vertices-per-triangle ratio. Try to get this value below one if possible.

2.28. Processing Load: Compute

What does this counter show?

This counter represents the average compute workload of the Shader Processor. This average is calculated across the currently selected time range, regardless whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where Compute tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent executing compute kernels.

If this value is high, then you should do the following:

- Profile with the offline compiler: You can batch process your application's compute kernels with the appropriate offline profiling compiler in the PowerVR SDK, which will help you find the best place to focus your optimization.

2.29. Processing Load: Pixel

What does this counter show?

This counter represents the average pixel workload of the Shader Processor. This average is calculated across the currently selected time range, regardless whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where Renderer tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent shading fragments.

If this value is high, then you should do the following:

- Profile with the offline compiler or PVRTrace: Use PVRTrace's Shader Analysis to isolate the most expensive shaders in your scene and use the built-in shader profiler to optimize them. Alternatively, you could batch process your application's shaders with the appropriate offline

profiling compiler in the PowerVR SDK using the "-perfsim" flag to identify the most expensive ones in your scene.

- Reduce alpha blending and discard/alpha test: The number of fragments given to the Shader Processor by the Texturing and Shading units is affected by the number of blended and alpha tested primitives that are being rendered. Because of this, reducing the amount of blending and discard in an application's render can reduce the Shader Processor's load.

2.30. Processing Load: Vertex

What does this counter show?

This counter represents the average vertex workload of the Shader Processor. This average is calculated across the currently selected time range, regardless whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where Tiler tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent shading vertices.

If this value is high, then you should do the following:

- Profile with the offline compiler or PVRTrace: Use PVRTrace's Shader Analysis to isolate the most expensive shaders in your scene and use the built-in shader profiler to optimize them. Alternatively, you could batch process your application's shaders with the appropriate offline profiling compiler in the PowerVR SDK using the "-perfsim" flag to identify the most expensive ones in your scene.
- Improve CPU object culling: Improving CPU culling of objects so there are fewer draw calls will reduce the Shader's vertex processing load, as fewer vertices will be submitted to the GPU. This culling should, ideally, be applied to geometry inside the view frustum as well as outside of it.
- Increase vertex sharing: Use index list and try to have a low vertices-per-triangle ratio. Try to get this value below 1 if possible.

2.31. Rays Emitted Per Second

What does this counter show?

This counter represents the average number of rays-emitted-per-second by the GPU over the selected period of time.

What does a high value mean?

A high value indicates that the GPU has been able to emit a high number of rays during the selected period of time.

2.32. Rays Per Second

What does this counter show?

This counter represents the average number of rays-processed-per-second by the GPU over the selected period of time.

What does a high value mean?

A high value indicates that the GPU has been able to process a high number of rays during the selected period of time.

2.33. Rays Per Task

What does this counter show?

This counter represents the average number of rays emitted from both ray and frame shaders.

What does a high value mean?

A high value indicates that on average a large number of rays are being emitted from ray and frame shaders. In isolation this number can be misleading and should instead be investigated alongside the

counters Rays Emitted per Second and Ray Tasks per Second to gain a better picture of the balance between the number of rays emitted and the number of ray shaders executing. An application could for example emit many rays for each ray or frame shader execution but may minimise the number of frame and ray shader executions. Conversely an application may emit few rays per frame or ray shader execution but may have many executing ray or frame shaders.

2.34. Ray Results Per Second

What does this counter show?

This counter represents the average number of rays-versus-triangle intersection results-per-second processed by the GPU over the selected period of time.

What does a high value mean?

A high value indicates that the GPU has carried out a high number of ray-versus-triangle-intersections during the selected period of time.

2.35. Ray Tasks Per Second

What does this counter show?

This counter represents the number of ray tasks processed per second.

What does a high value mean?

A high value indicates that a large number of ray or frame shader tasks have been processed.

If this value is high, then you should do the following:

- An attempt should be made to identify the highest priority shaders in the scene i.e. those which are both expensive to run and those that are being run frequently. The most expensive ray shader in the scene may very well not be the ray shader taking up the most Shader Processor time as it may not be executed very frequently. The highest priority ray shader can very often be a default ray shader used in a high proportion of ray emissions. It should be noted that any changes to the scene, or to the ray tracing techniques used may result in dramatic changes with regard to which shaders are executing most frequently. Therefore an attempt should be made to optimise all shaders to some extent.
- The highest priority ray shaders should then be profiled using the offline compiler.
- The number of ray emissions can also be reduced to reduce the frequency of ray shader executions.

2.36. Register Overload

What does this counter show?

This counter indicates when the hardware is under register pressure. Register pressure means we cannot queue as many tasks to the hardware due to register requirements being too high. This reduces latency and bandwidth tolerance because we have less tasks available to switch to - hiding these stalls.

What does a high value mean?

The value should be near 0% or very low in most situations. In some rare cases it might be higher, indicating that an "overload" has happened. Reducing queued tasks can increase latency, reduce available memory bandwidth and reduce performance.

2.37. Renderer Active

What does this counter show?

This counter shows the percentage of time that Renderer tasks were active. Renderer time refers to any time that is spent processing pixels and shading them. This includes the ISP (Image Synthesis Processor), Texturing and Shader Processor units.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of Renderer task timing data.

What does a high value mean?

A high value indicates that Renderer tasks have been active for a significant percentage of time.

If this value is high, you should determine if the bottleneck is caused by the Shader Processing or Texturing. Texture fetches and memory latency are the most common bottlenecks, followed by Shader Processing time. The ISP is rarely a bottleneck.

2.38. Renderer Time Per Frame

What does this counter show?

This counter shows the time spent processing Renderer tasks (in seconds) during the specified period. Renderer time refers to any time that is spent processing pixels and shading them. This includes the ISP (Image Synthesis Processor), Texturing and Shader Processor units.

2.39. RTU Active

What does this counter show?

This counter shows the percentage of time that RTU tasks were active. RTU time includes frame processing, ray processing, fixed function ray traversal tests and queries, and frame buffer accumulation.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of RTU timing block.

What does a high value mean?

A high value indicates that RTU tasks have been active for a significant percentage of time.

If this value is high, you should determine where in the ray tracing pipeline the bottleneck is being caused. The possible bottlenecks in the ray tracing pipeline are frame processing, ray processing, fixed function ray traversal tests and queries, and frame buffer accumulation. It is possible to be limited by many of the traditional rendering bottlenecks - mainly texture and buffer fetches as well as memory latency. Both ray and frame shader processing time can also be a very common bottleneck. Other common bottlenecks for ray tracing can include:

- Over emission of rays, -the use of large ray bounce limits,
- The use of highly detailed geometry,
- Component groups built from a large number of components,
- Using over generous scene extents for component groups,
- Excessive frame buffer accumulations,
- Over use of user specified ray data,
- Overly nesting component groups.

It should also be noted that, unlike in traditional rasterization, the combination as well as the complexity of objects within the scene can play a large part in the efficiency of the application. This is due to the way in which rays allow some amount of interaction between objects through the transmission of data using rays.

2.40. RTU Task Time Per Frame

What does this counter show?

This counter shows the time spent processing RTU tasks (in seconds) during the specified period. RTU time refers to any time taken to carry out frame processing, ray processing, fixed function ray traversal tests and queries, and frame buffer accumulation.

2.41. Shaded Pixels Per Frame

What does this counter show?

This counter represents the total number of pixels that the Shader unit has processed per frame. This includes the number of pixels visible and blended.

What does a high value mean?

A high value (for example, higher than the render target size) could indicate high overdraw and pixel processing load. In these cases pixel shaders should be made as simple as possible or the overdraw reduced, making alpha-blended polygons opaque.

2.42. Shaded Pixels Per Second

What does this counter show?

This counter represents the total number of pixels that the Shader unit has processed per second. This includes the number of pixels visible and blended.

What does a high value mean?

A high value (for example, higher than the render target size) could indicate high overdraw and pixel processing load. In these cases pixel shaders should be made as simple as possible or the overdraw reduced, making alpha-blended polygons opaque.

2.43. Shaded Vertices Per Frame

What does this counter show?

This counter represents the total number of vertices that the Shader unit has processed per frame. This includes all the vertices submitted by the user as any culling will happen after transformation.

What does a high value mean?

It is rare that a large value of processed vertices will have any effect on performance although, due to the fact that the Shader unit work-load is shared between vertex, pixel and compute tasks, it might represent extra pressure on the total processing load. Reducing the number of instructions per vertex should ease this pressure.

2.44. Shaded Vertices Per Second

What does this counter show?

This counter represents the total number of vertices that the Shader unit has processed per second. This includes all the vertices submitted by the user as any culling will happen after transformation.

What does a high value mean?

It is rare that a large value of processed vertices will have any effect on performance although, due to the fact that the Shader unit work-load is shared between vertex, pixel and compute tasks, it might represent extra pressure on the total processing load. Reducing the number of instructions per vertex should ease this pressure.

2.45. Shader Processing Load

What does this counter show?

This counter represents the average workload of the Shader Processor, i.e. when it is processing vertices, pixels or compute. This average is calculated across the currently selected time range, regardless whether the GPU is active or not. To get the most from this counter, you should inspect its value in periods where Vertex, Pixel or Compute tasks are active.

What does a high value mean?

A high value indicates that a large percentage of the Shader's workload has been spent processing vertices, fragments and/or compute kernels.

If this value is high, then you should refer to specific loads ("Processing load: Vertex", "Processing load: Pixel" & "Processing load: Compute") to see which type of processing load is bottlenecking your render.

2.46. SHF Bytes Per Triangle

What does this counter show?

This counter represents the average number of bytes per triangle in the SHF.

2.47. SHF Bytes Per Triangle (Varying)

What does this counter show?

This counter represents the average number of bytes per triangle (varyings) in the SHF.

2.48. SHF Bytes Per Triangle (Vertex)

What does this counter show?

This counter represents the average number of bytes per triangle (vertex) in the SHF.

2.49. SHF Triangles Per Second

What does this counter show?

This counter represents the average number of triangles processed by the Scene Hierarchy Generator front-end (SHF) over the selected period of time. The SHF unit is responsible for processing vertices and varyings and forwarding them onto the SHG unit as required.

What does a high value mean?

A high value indicates that the SHF unit has been able to process a high number of triangles during the selected period of time.

2.50. SHG Active

What does this counter show?

This counter shows the percentage of time that SHG tasks were active. SHG time includes ray vertex processing, primitive assembly and spatial index structure generation.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of SHG timing block.

What does a high value mean?

A high value indicates that SHG tasks have been active for a significant percentage of time.

If this value is high it indicates that a large percentage of time has been spent either carrying out vertex processing for scene components, carrying out primitive assembly or generating the spatial index structure. If this value is high then the following can be addressed by the user:

- Profile the ray vertex shaders with the offline compiler: It is possible to batch process your application's shaders with the appropriate offline profiling compiler in the PowerVR SDK. The "-profile" flag is used to identify the most expensive ones in the scene.
- Split the scene into a number of smaller separate blocks based on world space positions. Only build the blocks required for immediate shading, including both direct and indirect effects.
- Increase vertex sharing: Use index list to have a low vertices-per-triangle ratio. Try to get this value below one if possible.
- Reduce the number of vertices being processed within each build.

2.51. SHG Task Time Per Frame

What does this counter show?

This counter shows the time spent processing SHG tasks (in seconds) during the specified period. SHG time refers to any time taken to carry out ray vertex processing, primitive assembly and spatial index structure generation.

2.52. SHG Triangles Per Second

What does this counter show?

This counter represents the average number of triangles processed by the SHG over the selected period of time. The SHG unit takes the output of the SHF and is responsible for generating a scene hierarchy acceleration structure for the provided components which can later be used by the RTU.

What does a high value mean?

A high value indicates that the SHG unit has been able to process a high number of triangles during the selected period of time.

2.53. SH Load

What does this counter show?

This counter represents the average load of the GPU's Scene Hierarchy Generator units including Scene Hierarchy Generator Front-end(SHF) and Scene Hierarchy Generator Back-end(SHG) unit compared to their peak throughput. The SHF unit is responsible for processing vertices and varyings and forwarding them onto the SHG unit as required. The SHG unit takes the output of the SHF and is responsible for generating a scene hierarchy acceleration structure for the provided components, which can later be used by the Ray Tracing Unit (RTU).

A build command is used to initiate the building of a Scene Hierarchy. The first task in Scene Hierarchy Generation is the processing of vertices for each component in the component group where vertices and varyings are processed and forwarded to the SHG. The SHG then subdivides the triangles into voxels, and generates a spatial index structure which can later be used to accelerate ray processing.

What does a high value mean?

A high load value indicates that the SHF and SHG units are spending a significant amount of time processing vertices and varyings and generating the spatial index structure.

When the load is high, you should refer to the following counters for more information about the bottlenecks:

- SHF triangles per second,
- SHG triangles per second.

The total number of vertices submitted to the Scene Hierarchy Generator units should also be inspected.

2.54. Slot Occupancy

What does this counter show?

The percentage of time where the USC looked for a slot that is ready to run, and found one.

What does a high value mean?

A high value means that the USC is never starved; it always has something that can run.

A low value means that either there is no work to do, or that there is work to do but it is being stalled.

If this value is low at times when it should be high, check the Register Overload counter. It may be that additional slots are not ready to run, because running slots are using too many registers.

2.55. System Memory Load

What does this counter show?

This counter represents the current load on the system memory. It is the percentage of the total memory that is in use.

What does a high value mean?

If the Memory load is very high then the operating system is likely to either swap out memory or close background applications.

2.56. System Memory Total

What does this counter show?

This counter indicates the total amount of system memory available, in megabytes.

2.57. System Memory Use

What does this counter show?

This counter indicates the amount of system memory that is in use, in megabytes.

What does a high value mean?

If the amount of memory used is high relative to the amount of memory available, then the application is allocating too much. Runtime allocations may reduce performance.

If this value is high, then you should reduce memory allocations.

2.58. Texture Fetches Per Pixel

What does this counter show?

This counter shows the average number of texture requests per shaded pixel.

What does a high value mean?

A high value indicates that, on average, there are a large number of texture fetch operations in fragment shaders.

When the load is high, you should refer to the following counters for more information about the bottleneck:

- GPU Memory Write Rate (Mbytes/sec) and GPU Memory Read Rate (Mbytes/sec): If these values are high, the large number of texture fetches may be causing a system memory bottleneck
- Texturing Load (%): If this load is high, the large number of texture fetch requests has caused a Texture Unit bottleneck
- Texture Overload (%): If the number of texture fetches per-pixel is very high, it is likely that texture overload events will occur. These events can reduce Active Slot Occupancy, in turn reducing the Shader Processor's ability to hide latency caused by data dependencies

2.59. Texture Overload

What does this counter show?

This counter shows when texture overload events have occurred. Each event indicates that a Texture sample request queue is full, i.e. shader processing units are submitting requests faster than the Texturing Unit can process them.

What does a high value mean?

When a texture overload event occurs, the Shader Processor has to de-schedule any tasks that have not been able to submit work to the Texturing Unit queue. The Shader Processor will then try to schedule ready tasks from Available Slots into Active Slots. If there are no ready tasks to schedule (i.e. all Available Slots are waiting to resolve data dependencies), the Shader Processor will not be able to fill the vacant Active Slots. This will result in reduced Active Slot occupancy.

When the load is high, you should refer to the following counters for more information about the bottleneck:

- Active Slot Occupancy (%): A low value indicates that the Texturing Unit blocked queue has caused a reduction of in-flight Shader Processor tasks. The lower the number of in-flight tasks (Active Slots), the less opportunity the Shader Processor has to hide latency caused by data dependencies.

2.60. Texturing Load

What does this counter show?

This counter represents the average load of Texturing Units compared to their peak throughput.

What does a high value mean?

A high value (e.g. beyond 50%) indicates that the Texturing Units are spending a significant amount of time fetching texture data from system memory and/or performing linear interpolation filtering operations.

When the load is high, you should refer to the following counters for more information about the bottleneck:

- Texture Fetches per Pixel: A high value suggests that the number of samples requested by shaders is causing a bottleneck,
- Texture Overload (%): If Texturing Load is high it is likely that texture overload events will occur. These events can reduce Active Slot Occupancy, in turn reducing the Shader Processor ability to hide latency caused by data dependencies.

2.61. Tiler Active

What does this counter show?

This counter shows the percentage of time that Tiler tasks were active. Tiler time includes vertex processing and all the projection, clipping, culling and tiling/binning operations.

It is worth noting that this counter is effectively a Boolean value that shows this processing stage as either active or idle. The counter is best used to graph activity over a long period of time. In small periods, the counter value will match the activity of Tiler timing block.

What does a high value mean?

A high value indicates that Tiler tasks have been active for a significant percentage of time. This may be caused by the Tiler processing a large number of polygons and/or performing a lot of tiling work, e.g. many large primitives each touching many tiles.

There are very few cases where Tiler tasks are likely to be the bottleneck. If the value is very high (e.g. >80%) over a sustained period (e.g. one second), then you should try the following to reduce Tiler time:

- Improve CPU object culling: Improving CPU culling of objects so there are fewer draw calls will reduce the Tiler load as fewer polygons will be submitted to the GPU and tiling work may also reduce. This culling should, ideally, be applied to geometry inside the view frustum as well as outside of it.
- Reduce polygon count/tessellation factor: Reducing the number of polygons that are submitted to the GPU can reduce the Tiler load, as less clipping and culling operations will be required. Polygon/tessellation reduction should, ideally, be applied to geometry inside the view frustum as well as outside of it.

2.62. Tiler Time Per Frame

What does this counter show?

This counter shows the time spent processing Tiler tasks (in seconds) during the specified period. Tiler time includes vertex processing and all the projection, clipping, culling and tiling/binning operations.

2.63. Tiles Per Triangle

What does this counter show?

The average number of tiles a triangle touches. This counter gives an indication of how big the triangles are in the scene when projected to screen-space and in how many tiles will need to be processed for each triangle.

What does a high value mean?

It indicates that there are large triangles in the scene which might incur in a high load in the HSR processing and Parameter Buffer storage area.

If this value is high, then you should do the following:

- Avoid non visible screen-sized objects: At very high resolutions simple objects like a background or a sky-box may cause reduced performance. If this is a bottleneck in your render, try to reduce the number of screen-sized objects. If the objects cannot be removed, then try to reduce the screen-space size of your primitives (e.g. increase the number of polygons used to represent those objects).

2.64. Triangles Culled

What does this counter show?

The percentage of post-transform triangles culled before data is written to the Parameter Buffer. These culled triangles include sub-pixel, back-face and off-screen polygons.

What does a low value mean?

A low value means that not many polygons are being culled, which could cause more data to be written to the Parameter Buffer, in turn increasing the memory bandwidth cost of the render.

If this value is low, then you should do the following:

- Enable back-face culling: Enable this simple operation when possible as it could reject half of the polygons submitted for standard meshes.
- Perform culling on the CPU: Removing objects that are outside of the view frustum or occluded by opaque geometry will reduce the amount of data written to the Parameter Buffer by the GPU.

2.65. Triangles Per Draw

What does this counter show?

This counter represents the average number of triangles submitted to the Tiler per draw.

What does a high value mean?

A high value indicates that, on average, many triangles are submitted by each draw call. If this value is high, then you should do the following:

- Reduce geometric complexity of the scene: Reduce the number of triangles used to represent each object

2.66. Triangles Input Per Frame

What does this counter show?

This counter represents the total number of triangles submitted to the Tiler per frame.

What does a high value mean?

If you are familiar with your target device's peak triangle throughput, this counter will give an at-a-glance understanding of how close you are to peak and if the number of triangles submitted needs to be reduced. If the value is high, you should:

- Reduce polygon count: Reduce the number of triangles used to represent the objects in your scene. Remember that objects further in the background can be significantly simpler without sacrificing perceived detail for the user.

2.67. Triangles Input Per Second

What does this counter show?

This counter represents the total number of triangles submitted to the Tiler per second.

What does a high value mean?

If you are familiar with your target device's peak triangle throughput, this counter will give an at-a-glance understanding of how close you are to peak and if the number of triangles submitted needs to be reduced. If the value is high, you should:

- Reduce polygon count: Reduce the number of triangles used to represent the objects in your scene. Remember that objects further in the background can be significantly simpler without sacrificing perceived detail for the user

2.68. Triangles Output Per Frame

What does this counter show?

This counter represents the total triangles written to the Parameter Buffer per-frame after back-face, off-screen and sub-pixel culling.

What does a high value mean?

A value close to "Triangles input per frame" indicates that very little GPU culling has been performed. A high value may mean that a large amount of data is being written to the Parameter Buffer every frame. A high value will also increase ISP load, as more geometry will have to be processed per-tile.

If this value is high, then you should do the following:

- Enable back-face culling when possible: Do not submit triangles unnecessarily to the GPU,
- Perform culling by-hand on the CPU: Do not submit triangles unnecessarily to the GPU,
- Reduce geometry complexity: Simplify models and use appropriate Level-of-detail techniques.

2.69. Triangles Output Per Second

What does this counter show?

This counter represents the total triangles written to the Parameter Buffer per-second after back-face, off-screen and sub-pixel culling.

What does a high value mean?

A value close to "Triangles input per second" indicates that very little GPU culling has been performed. A high value may mean that a large amount of data is being written to the Parameter Buffer every second. A high value will also increase ISP load, as more geometry will have to be processed per-tile.

If this value is high, then you should do the following:

- Enable back-face culling when possible: Do not submit triangles unnecessarily to the GPU,
- Perform culling by-hand on the CPU: Do not submit triangles unnecessarily to the GPU,
- Reduce geometry complexity: Simplify models and use appropriate Level-of-detail techniques.

2.70. Triangle Ratio

What does this counter show?

This counter represents ratio of triangles output from the Tiler over triangles input to the Tiler.

Triangle culling can remove triangles and reduces the ratio, whilst triangle clipping can add triangles and increases the ratio.

2.71. Vertices Per Triangle

What does this counter show?

Average number of vertices per triangle. This is calculated as the number of input vertices processed divided by the number of input triangles processed. This counter gives an indication of how efficiently transformed vertices are shared between triangles.

What does a high value mean?

This value varies between a maximum of 3, which indicates that there is no sharing and every triangle has an individual index per vertex, to a number close to or below 1. The lower this number is, the most optimal the geometry is for processing.

If this value is high:

- Improve triangle sorting for meshes: For optimal performance, triangles should be sorted by spacial locality to improve post-transform vertex cache efficiency. Our PVRGeoPOD exporter performs triangle sorting by default. Our triangle sorting code can also be found in our SDK Framework.

2.72. Z Load Store

What does this counter show?

For most renders, depth and stencil buffers only contain temporary data required to complete the associated render pass. In the PowerVR architecture, on-chip depth and stencil buffers are used to store this data. When the appropriate API mechanisms are utilised (please refer to our Performance Recommendations document for more information), an application will never cause data to be uploaded to or written from these on-chip buffers. This enables an application to avoid redundant system memory transfers for these temporary buffers.

A Z load store event indicates that there has been an upload or resolve of depth/stencil data to/from the GPU. Unless a given application requires depth or stencil information to be preserved, an application should always use the appropriate API mechanisms to avoid these costly data transfer operations.

What does a high value mean?

A value above 0% indicates a Z load store event has occurred. To avoid these operations, you should ensure depth and stencil buffers that do not need to be preserved are cleared at the start and invalidated at the end of each render pass.

3. Additional Counters

These counters will only appear when PVRTrace is used in conjunction with PVRTune.

3.1. Buffer Object Modifications (bytes) Per Frame

What does this counter show?

This counter shows the buffer object modifications (bytes) per frame.

3.2. Buffer Object Modifications (bytes) Per Second

What does this counter show?

This counter shows the buffer object modifications (bytes) per second.

3.3. Buffer Object Uploads (bytes) Per Frame

What does this counter show?

This counter shows the buffer object uploads (bytes) per frame.

3.4. Buffer Object Uploads (bytes) Per Second

What does this counter show?

This counter shows the buffer object uploads (bytes) per second.

3.5. Compute Shader Compiles Per Frame

What does this counter show?

This counter shows the compute shader compiles per frame.

3.6. Compute Shader Compiles Per Second

What does this counter show?

This counter shows the compute shader compiles per second.

3.7. Context Binds Per Frame

What does this counter show?

This counter shows the context binds per frame.

3.8. Context Binds Per Second

What does this counter show?

This counter shows the context binds per second.

3.9. Fragment Shader Compiles Per Frame

What does this counter show?

This counter shows the fragment shader compiles per frame.

3.10. Fragment Shader Compiles Per Second

What does this counter show?

This counter shows the fragment shader compiles per second.

3.11. Frame Buffer Accesses Per Frame

What does this counter show?

This counter shows the frame buffer accesses per frame.

3.12. Frame Buffer Accesses Per Second

What does this counter show?

This counter shows the frame buffer accesses per second.

3.13. Framebuffer Access (bytes) Per Frame

What does this counter show?

This counter shows the frame buffer access per frame.

3.14. Framebuffer Access (bytes) Per Second

What does this counter show?

This counter shows the frame buffer accesses per second.

3.15. Indexed Draw Calls Per Frame

What does this counter show?

This counter shows the indexed draw calls per frame.

3.16. Indexed Draw Calls Per Second

What does this counter show?

This counter shows the indexed draw calls per second.

3.17. Line no. (list) Per Frame

What does this counter show?

This counter shows the number of lines submitted by GL_LINES per frame.

3.18. Line no. (list) Per Second

What does this counter show?

This counter shows the number of lines submitted by GL_LINES per second.

3.19. Line no. (loop) Per Frame

What does this counter show?

This counter shows the number of lines submitted by GL_LINE_LOOPs per frame

3.20. Line no. (loop) Per Second

What does this counter show?

This counter shows the number of lines submitted by GL_LINE_LOOPs per second.

3.21. Line no. (strip) Per Frame

What does this counter show?

This counter shows the number of lines submitted as GL_LINE_STRIPs per frame.

3.22. Line no. (strip) Per Second

What does this counter show?

This counter shows the number of lines submitted as GL_LINE_LOOPs per second.

3.23. Non-Indexed Draw Calls Per Frame

What does this counter show?

This counter shows the number of non-indexed draw calls per frame.

3.24. Points no. Per Frame

What does this counter show?

This counter shows the number of GL_POINTs submitted per frame.

3.25. Points no. Per Second

What does this counter show?

This counter shows the number of GL_POINTs submitted per second.

3.26. Program Links Per Frame

What does this counter show?

This counter shows the number of shader programs that are linked per frame.

3.27. Program Links Per Second

What does this counter show?

This counter shows the number of shader programs that are linked per second.

3.28. Scissor Calls Per Frame

What does this counter show?

This counter shows the number of scissor modifications made per frame.

3.29. Scissor Calls Per Second

What does this counter show?

This counter shows the number of scissor modifications made per second.

3.30. Texture Modifications (bytes) Per Frame

What does this counter show?

This counter shows the amount of texture modifications per frame in bytes.

3.31. Texture Modifications (bytes) Per Second

What does this counter show?

This counter shows the amount of texture modifications per second in bytes.

3.32. Texture Modifications Per Frame

What does this counter show?

This counter shows the total number of texture modifications per frame.

3.33. Texture Modifications Per Second

What does this counter show?

This counter shows the total number of texture modifications per second.

3.34. Texture Uploads (bytes) Per Frame

What does this counter show?

This counter shows the amount of texture uploads per frame in bytes.

3.35. Texture Uploads (bytes) Per Second

What does this counter show?

This counter shows the amount of texture uploads per second in bytes.

3.36. Texture Uploads Per Frame

What does this counter show?

This counter shows the total number of texture uploads per frame.

3.37. Texture Uploads Per Second

What does this counter show?

This counter shows the total number of texture uploads per second.

3.38. Total API Calls Per Frame

What does this counter show?

This counter displays the total number of API calls per frame.

3.39. Total API Calls Per Second

What does this counter show?

This counter shows the total number of API calls per second.

3.40. Total Line no. Per Frame

What does this counter show?

This counter shows the total number of lines submitted per frame.

3.41. Total Line no. Per Second

What does this counter show?

This counter displays the total number of lines submitted per second.

3.42. Total Triangle no. Per Frame

What does this counter show?

This counter shows the total number of triangles submitted per frame.

3.43. Total Triangle no. Per Second

What does this counter show?

This counter shows the total number of triangles submitted per second.

3.44. Triangle no. (fan) Per Frame

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLE_FANs per frame.

3.45. Triangle no. (fan) Per Second

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLE_FANs per second.

3.46. Triangle no. (list) Per Frame

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLEs per frame.

3.47. Triangle no. (list) Per Second

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLEs per second.

3.48. Triangle no. (strip) Per Frame

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLE_STRIPs per frame.

3.49. Triangle no. (strip) Per Second

What does this counter show?

This counter shows the number of triangles submitted as TRIANGLE_STRIPs per seconds.

3.50. Uniform Uploads Per Frame

What does this counter show?

This counter shows the number of shader program uniform modifications per frame.

3.51. Uniform Uploads Per Second

What does this counter show?

This counter shows the number of shader program uniform modifications per second.

3.52. Vertex Shader Compiles Per Frame

What does this counter show?

This counter shows the number of vertex shader compilations per frame.

3.53. Vertex Shader Compiles Per Second

What does this counter show?

This counter shows the number of vertex shader compilations per second.

3.54. Viewport Calls Per Frame

What does this counter show?

This counter shows the number of viewport modifications made per frame.

3.55. Viewport Calls Per Second

What does this counter show?

This counter shows the number of viewport modifications made per second.

4. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp