Unreal Engine 4: Mobile Graphics on ARM CPU and GPU Architecture

Jesse Barker, Principal Software Engineer, ARM
Marius Bjørge, Graphics Research Engineer, ARM
Niklas "Smedis" Smedberg, Senior Engine Programmer, Epic Games
Brad Grantham, Principal Software Engineer, ARM
Graham Hazel, Senior Product Manager, Geomerics

Agenda

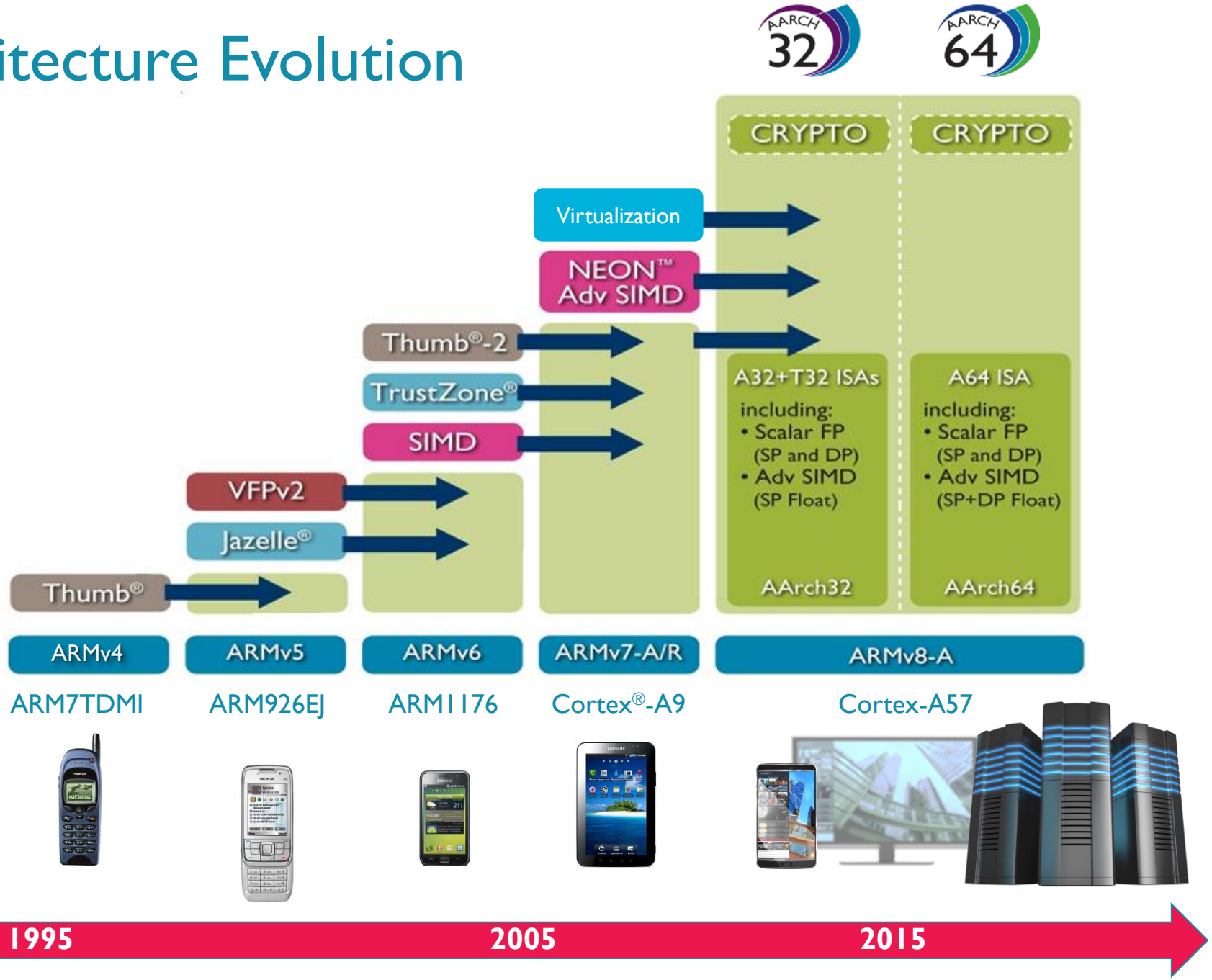
- Programming for ARM®v8-A Technology
- ARM Mali™ GPU Architecture
  - Hardware evolution
  - The tri-pipe architecture
  - Exposing the tile
- Unreal Engine 4 Case Study: Moon Temple
- Enlighten in Unreal Engine 4

ARM

Programming for ARMv8-A Technology

Jesse Barker
Principal Software Engineer, ARM

ARM Architecture Evolution

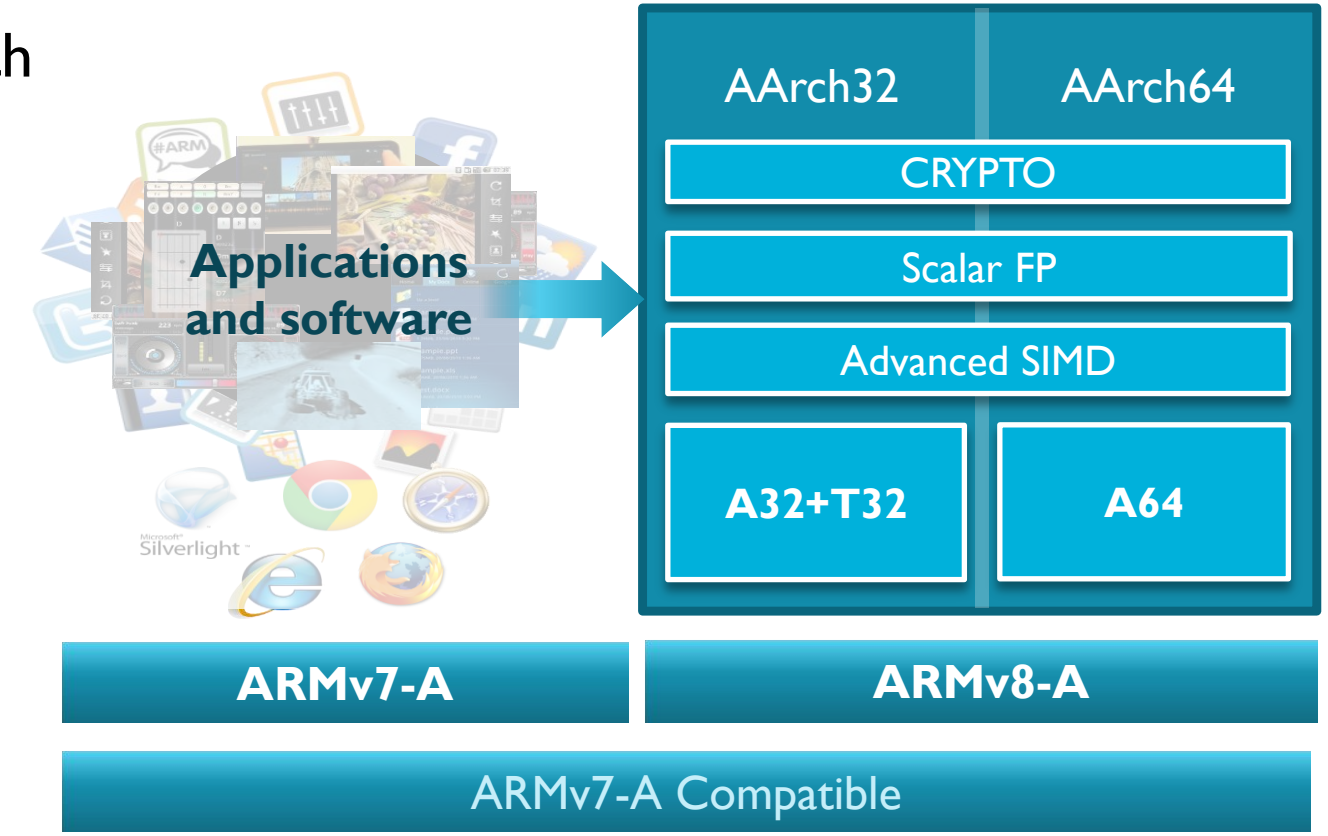


ARMv8-A AArch32

Maintaining compatibility

- AArch32 maintains full-compatibility with ARMv7 while addressing emerging software trends

- AArch32: evolution of 32-bit
  - Enhanced floating point support (IEE754-2008)
  - Ideal for concurrent programming
    C11, C++ 11, Java5
  - More efficient, high-performance thread-safe software
  - Cryptography support (AES, Sha-1, Sha-256)



ARMv8-A Architecture

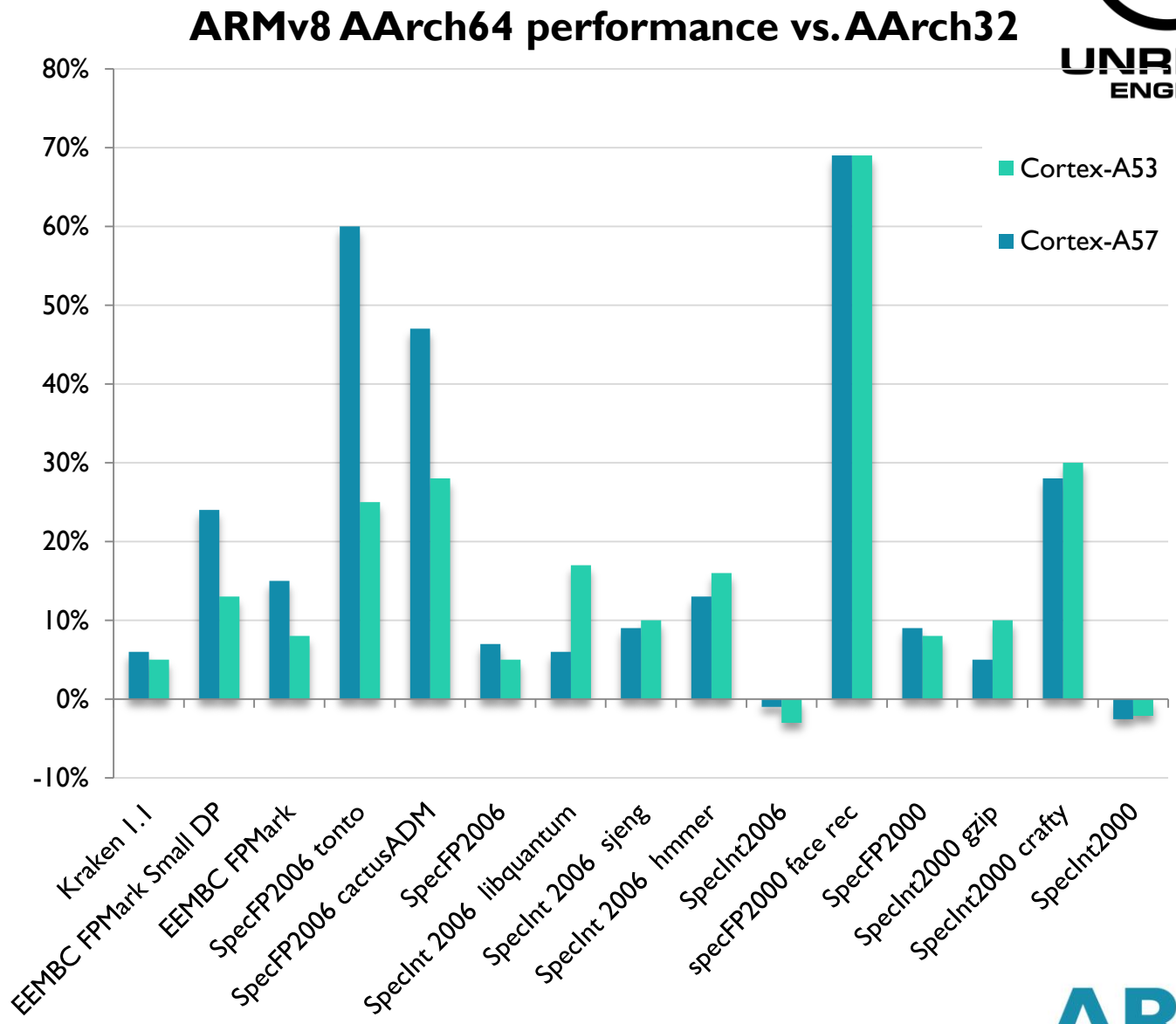
Designed for efficiency

Design	Why it Matters
64-bit architecture	Efficient access to large datasets
Increased number and size of general purpose registers	Gains in performance and code efficiency
Large Virtual Address Space	1. Applications not limited to 4GB memory 2. Large memory mapped files handled efficiently
Efficient 32-bit/64-bit architecture	1. Common software architecture (phone, tablet, clamshell) 2. A single software model across the entire portfolio
Double the number and size of NEON™ registers	Enhanced capacity of SIMD multimedia engine
Cryptography support	1. Over10x software encryption performance 2. New security models for consumer and enterprise

ARM

AArch64 Performance Over AArch32

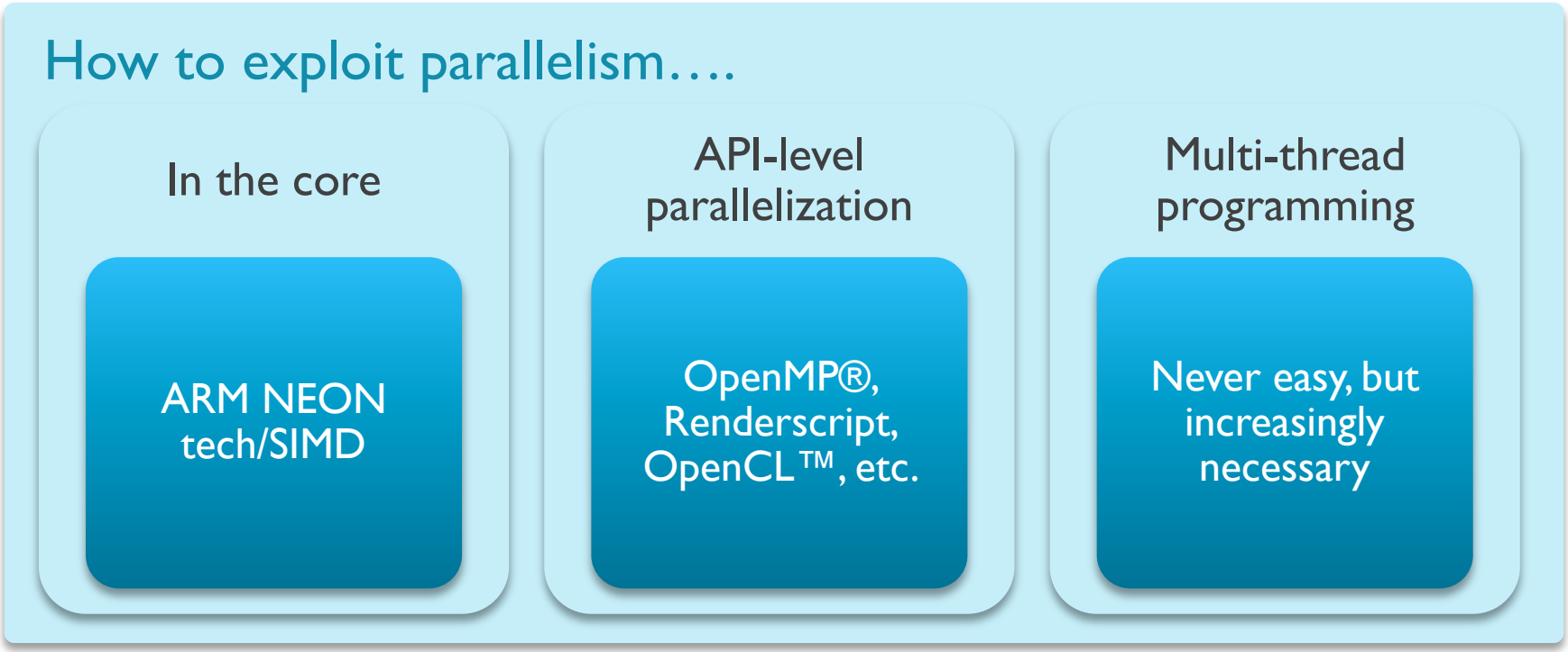
- **>20% increase** on several key workloads
- Most workloads increase, some slow down
  - Slowdowns are often outliers like mcf in spec2k with unrealistic data access patterns
- Overall trend is increasing performance with 64b
  - Will increase further as compilers mature



Multi-core ARM big.LITTLE™ Technology

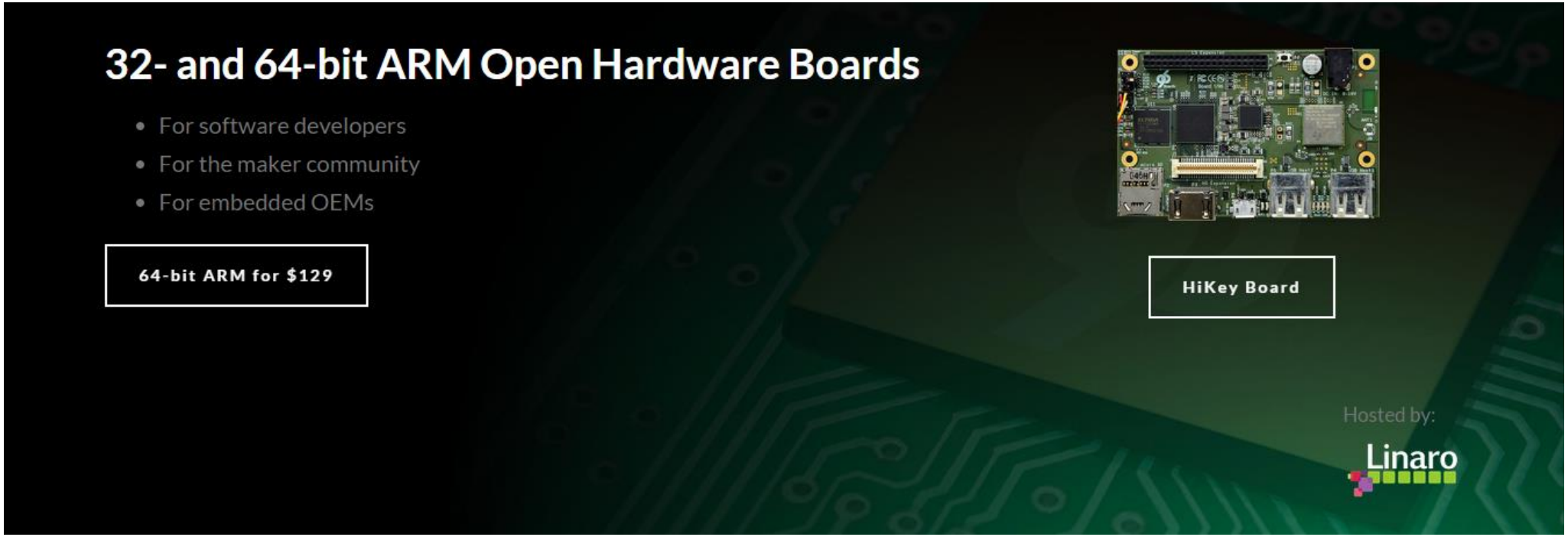
Taking advantage of parallelism

- Platform trending toward multi-cores
  - Single thread performance improvements diminishing
  - Thermally constrained use cases are now commonplace
  - Production differentiation via different CPU combinations
- Modern OSs are supporting multi-core



ARMv8-A and 64-bit Everywhere

Mega trend is the move to ARMv8-A and AARCH64



- Low cost development platforms available from 96boards.org
- Huge growth in share of 64-bit platforms in smartphone and tablets in 2015

Mali GPU Architecture

Marius Bjørge
Graphics Research Engineer, ARM

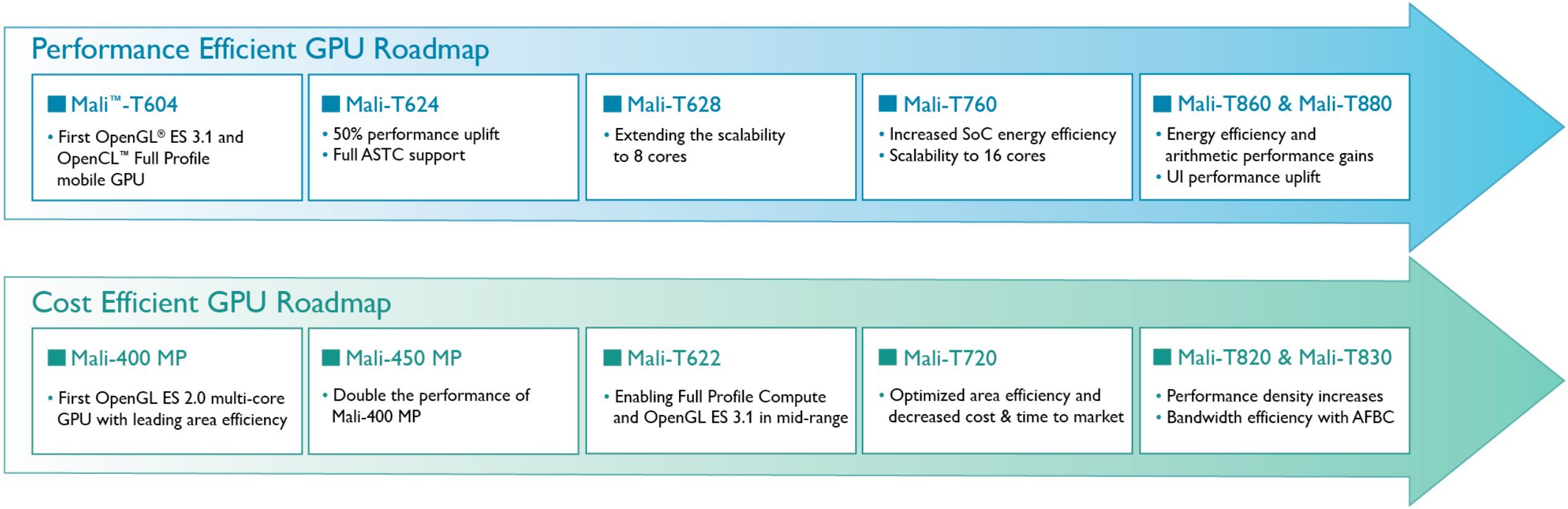
The Midgard Architecture

HARDWARE EVOLUTION

ARM

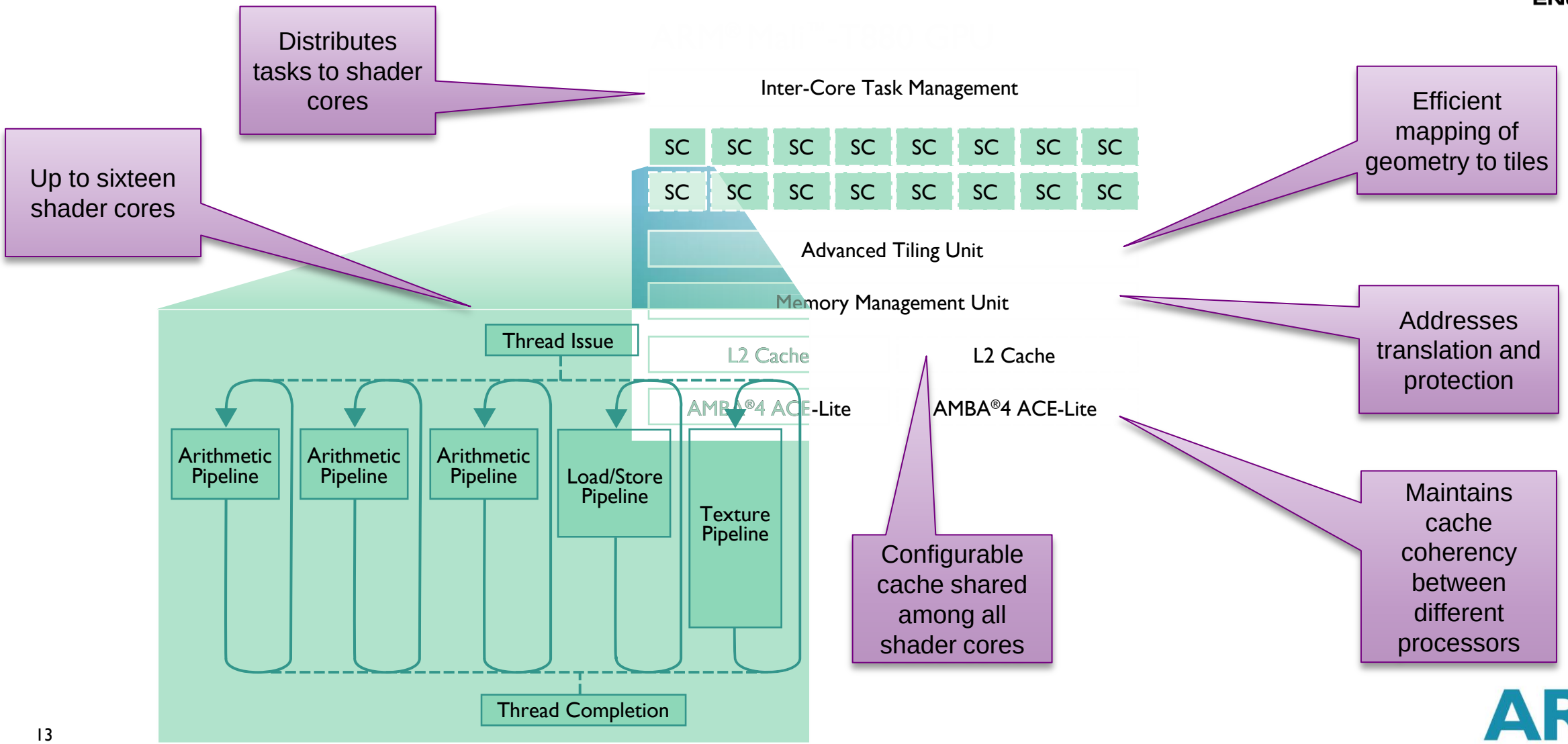
Driving for Efficiency

The Mali GPU roadmap



Mali GPU High-Level Architecture

A breakdown of the Mali-T880

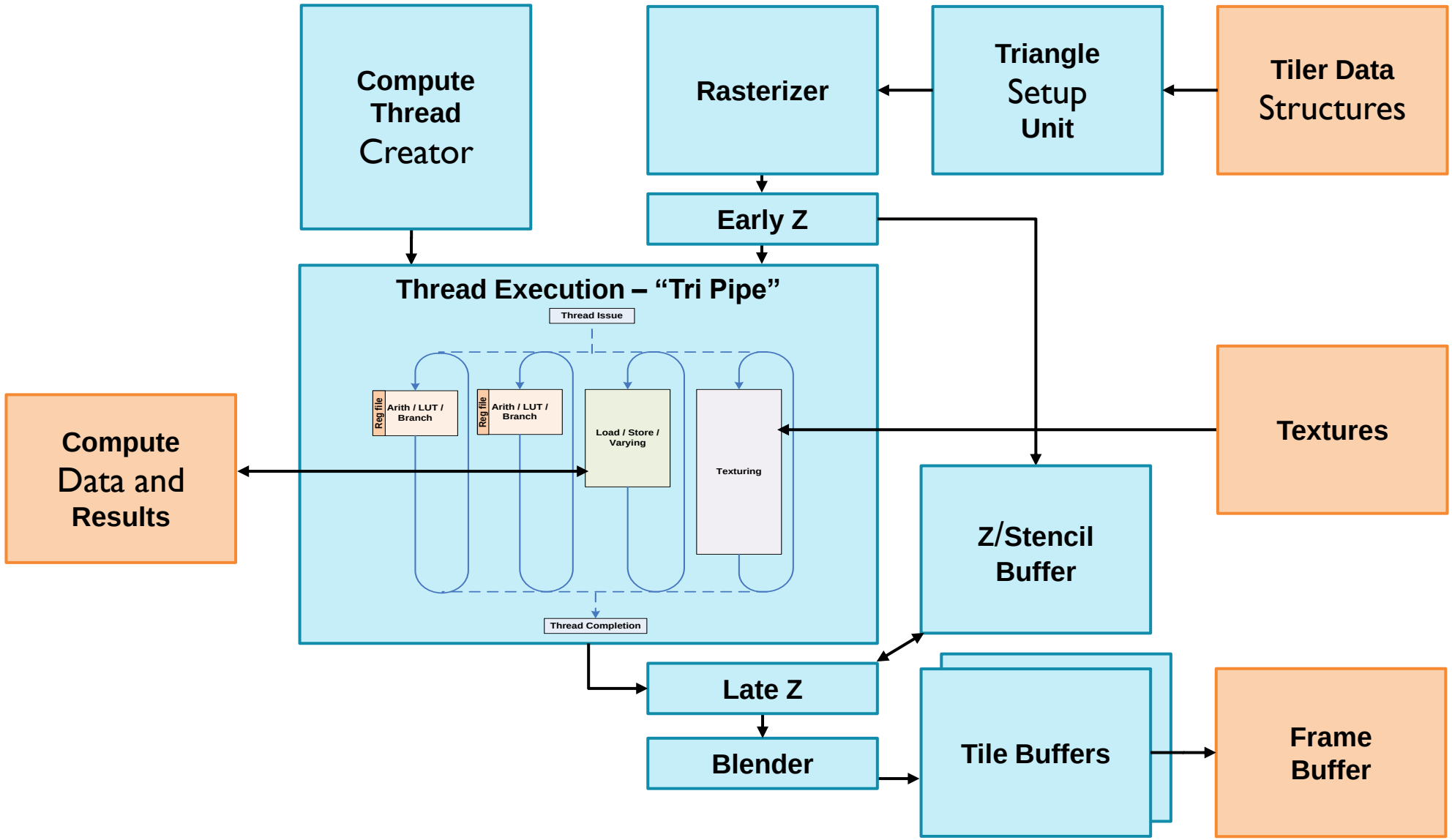


The Midgard Architecture

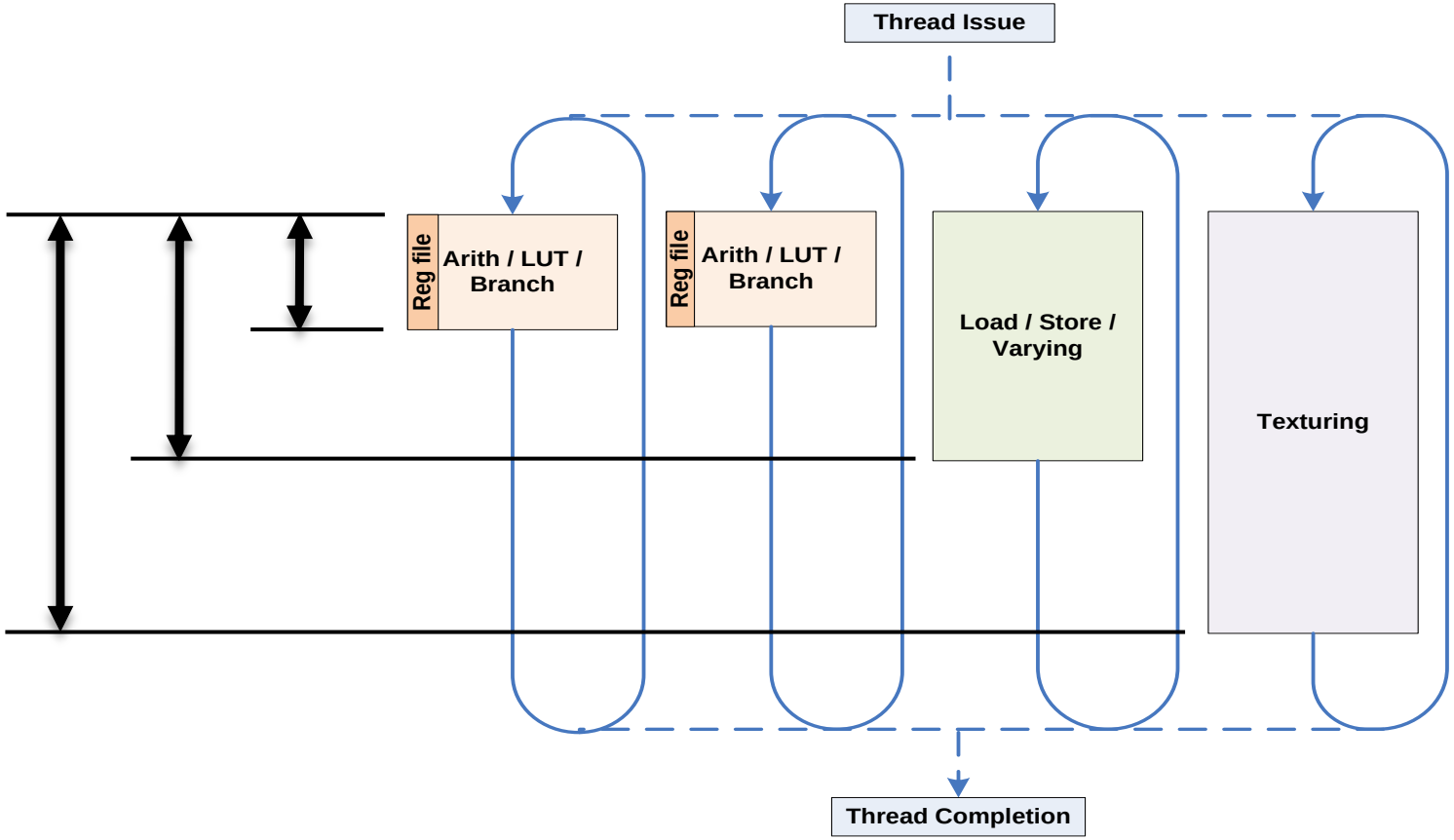
THE TRI-PIPE ARCHITECTURE

ARM

Shader Core Architecture



Tri-pipe Architecture



- Unified shader architecture
  - Fragment and vertex shaders
  - Compute shaders
- Very high throughput graphics
- Multiple parallel pipelines
  - Two low-latency arithmetic pipes
  - 256 simultaneous threads
  - Low-latency for computation

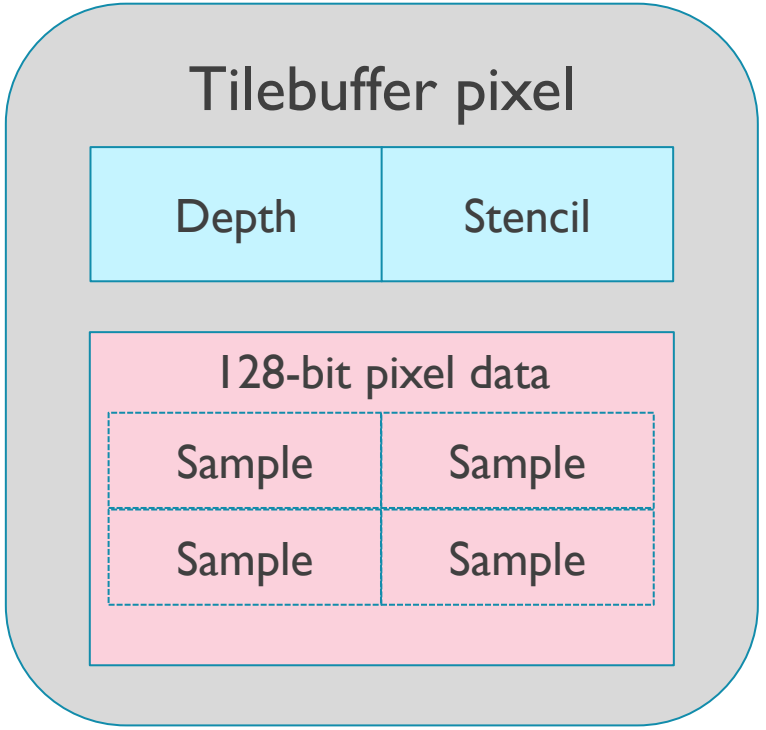
The Midgard Architecture

EXPOSING THE TILE

ARM

The Tilebuffer

- Mali-T600/T700/T800 Series GPU
  - Tile-based rendering
- 16x16 tile size
  - Fast on-chip memory
  - 16 bytes of per-pixel color data
  - Raw bit access
- More recent GPU architectures allow more flexible tile sizes and open up more per-pixel color data



Exposing the Tilebuffer

- Shader Framebuffer Fetch
  - Access previous fragment color, depth and stencil
  - Programmable blending, soft particles, etc.
- Shader Pixel Local Storage (PLS)

ARM

Pixel Local Storage (PLS)

- Exposed as EXT_shader_pixel_local_storage
- Per-pixel scratch memory available to fragment shaders
  - Automatically discarded once a tile is fully processed
  - No impact on external memory bandwidth
- Shader declares an interface block of PLS memory
  - Re-interpret PLS between different passes
  - Can have separate input and output views
  - Independent of framebuffer format

ARM

Pixel Local Storage

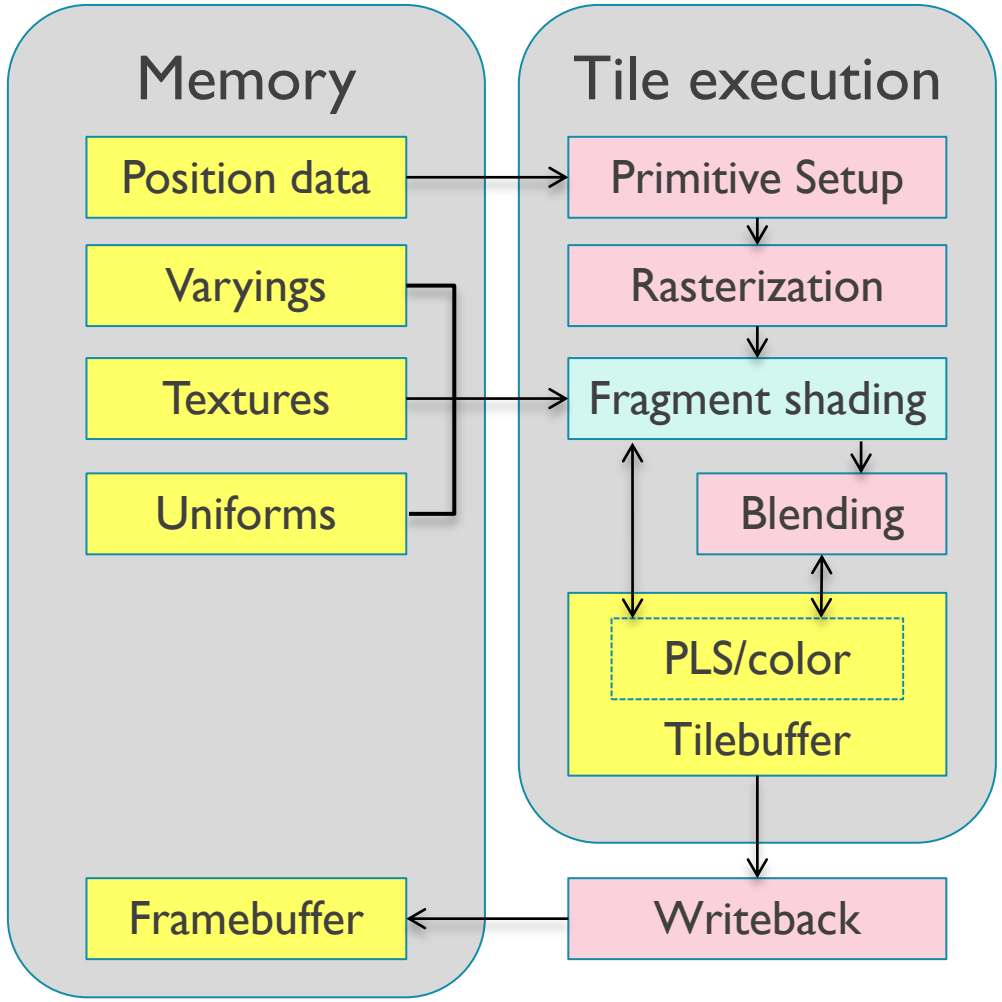

```
__pixel_localEXT FragDataLocal
{
  layout(r32f) highp float_value;
  layout(r11f_g11f_b10f) mediump vec3 normal;
  layout(rgb10_a2) highp vec4 color;
  layout(rgba8ui) mediump uvec4 flags;
} pls;
```

- See the extension spec for more information!
  - https://www.khronos.org/registry/gles/extensions/EXT/EXT_shader_pixel_local_storage.txt
  - http://malideveloper.arm.com

ARM

Pixel Local Storage

- Rendering pipeline changes slightly when PLS is enabled
  - Writing to PLS bypasses blending
- Note
  - Fragment order
  - Fragment tests still apply
  - PLS and color share the same memory location



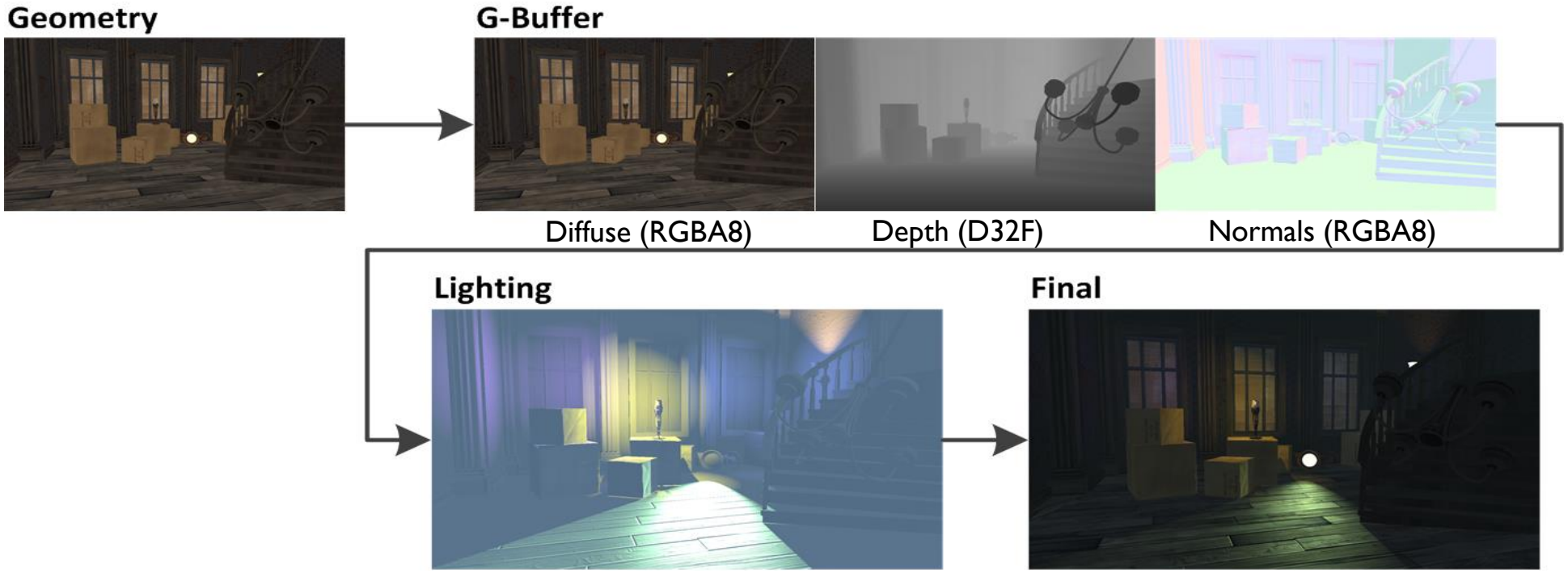
Why Pixel Local Storage?

- An alternative approach is to use multiple render targets (MRT) with framebuffer fetch
  - …if the driver can prove that render targets are not used later, it can avoid the write-back
- PLS is more explicit than MRT
  - Harder for the application to get it wrong
  - Driver doesn't have to make guesses
- PLS is more flexible
  - Re-interpret PLS data between fragment shader invocations
  - Not limited to OpenGL® ES 3.x framebuffer formats

ARM

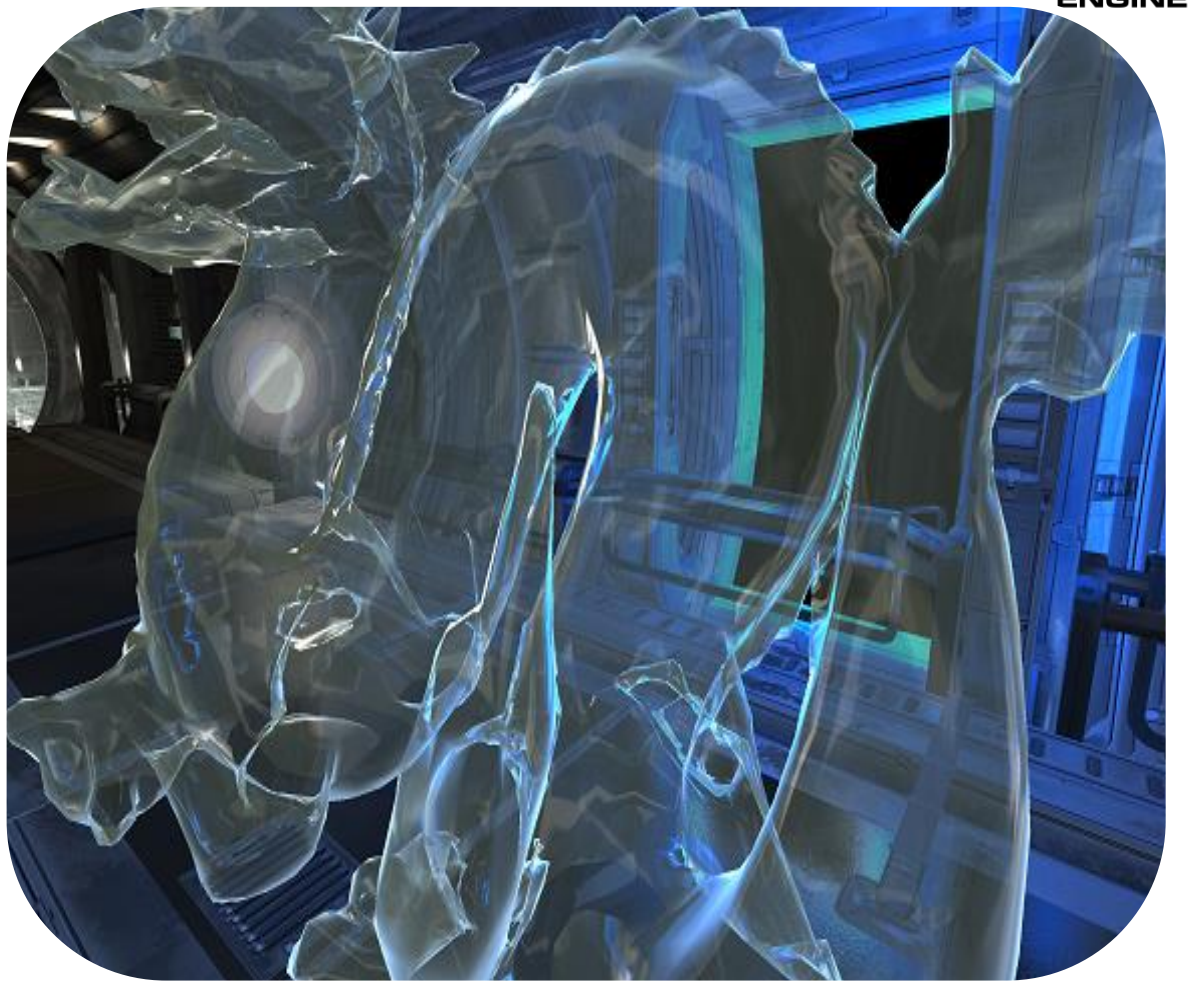
Deferred Shading

- Popular technique in PC and console games
  - Very memory bandwidth intensive
  - Traditionally not a good fit for mobile

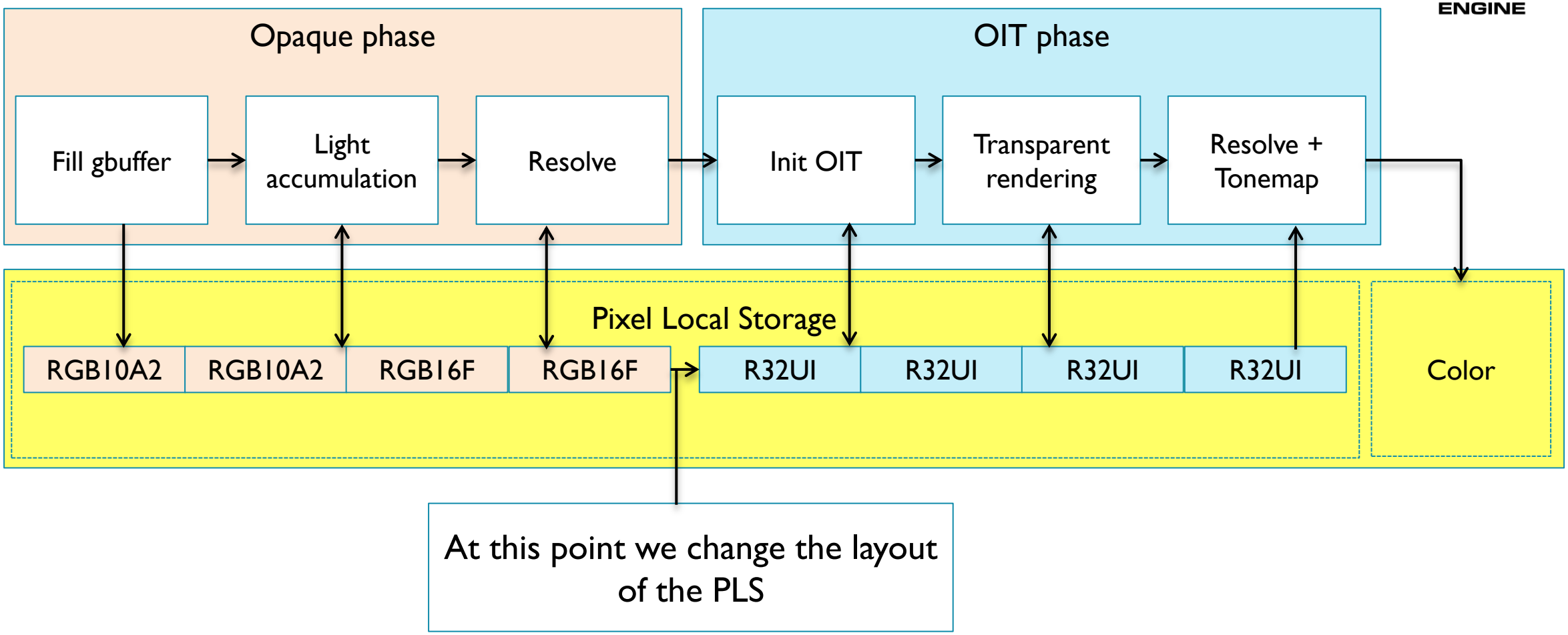


Order Independent Transparency

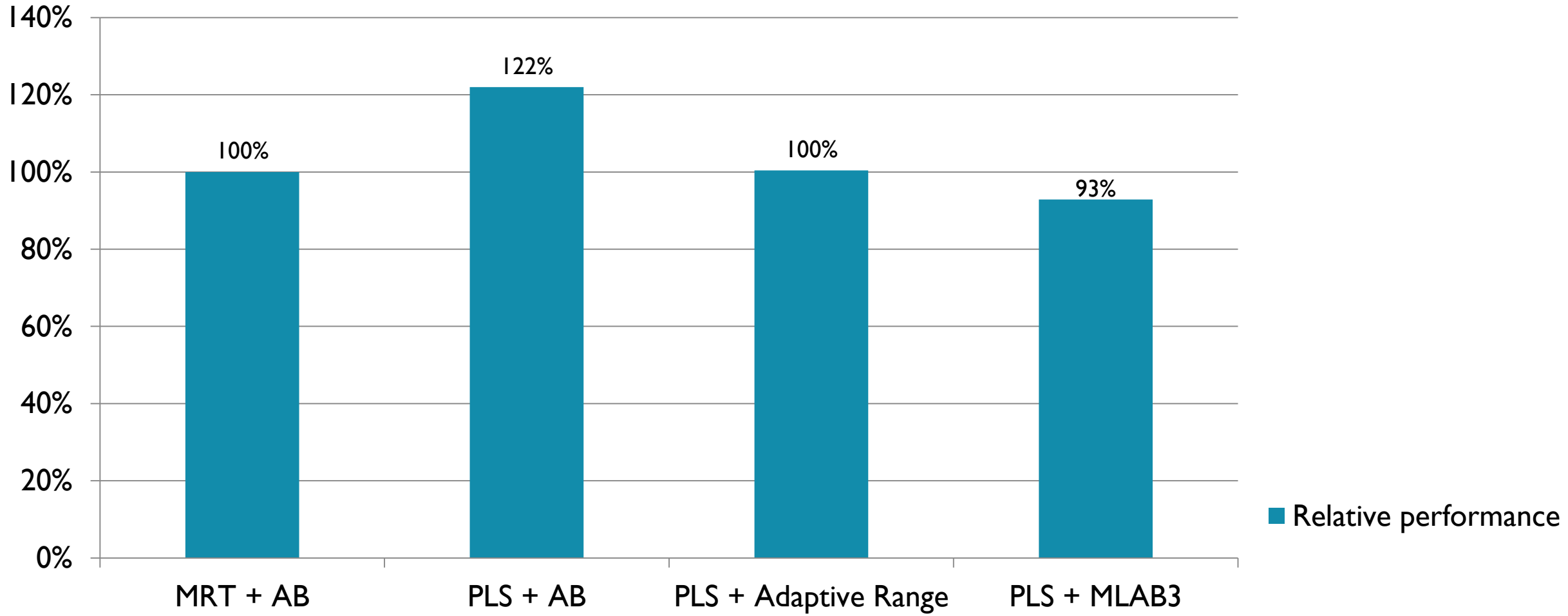
- "Unsolved" problem
- Depth peeling
- Approximate approaches
  - Multi-Layer Alpha Blending [Salvi et al, 2014]
  - Adaptive Range



Pixel Local Storage



Performance Comparison of Approaches



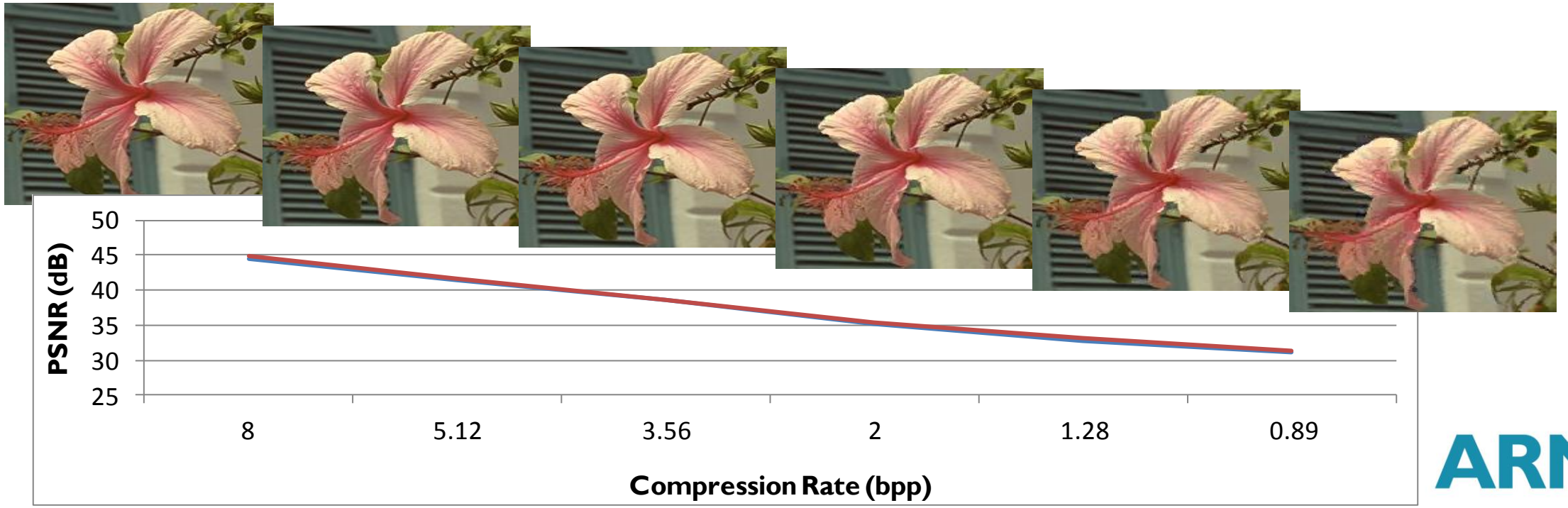
AB = Alpha Blending
MLAB3 = 3 layer Multi-Layer Alpha Blending

Unreal Engine 4

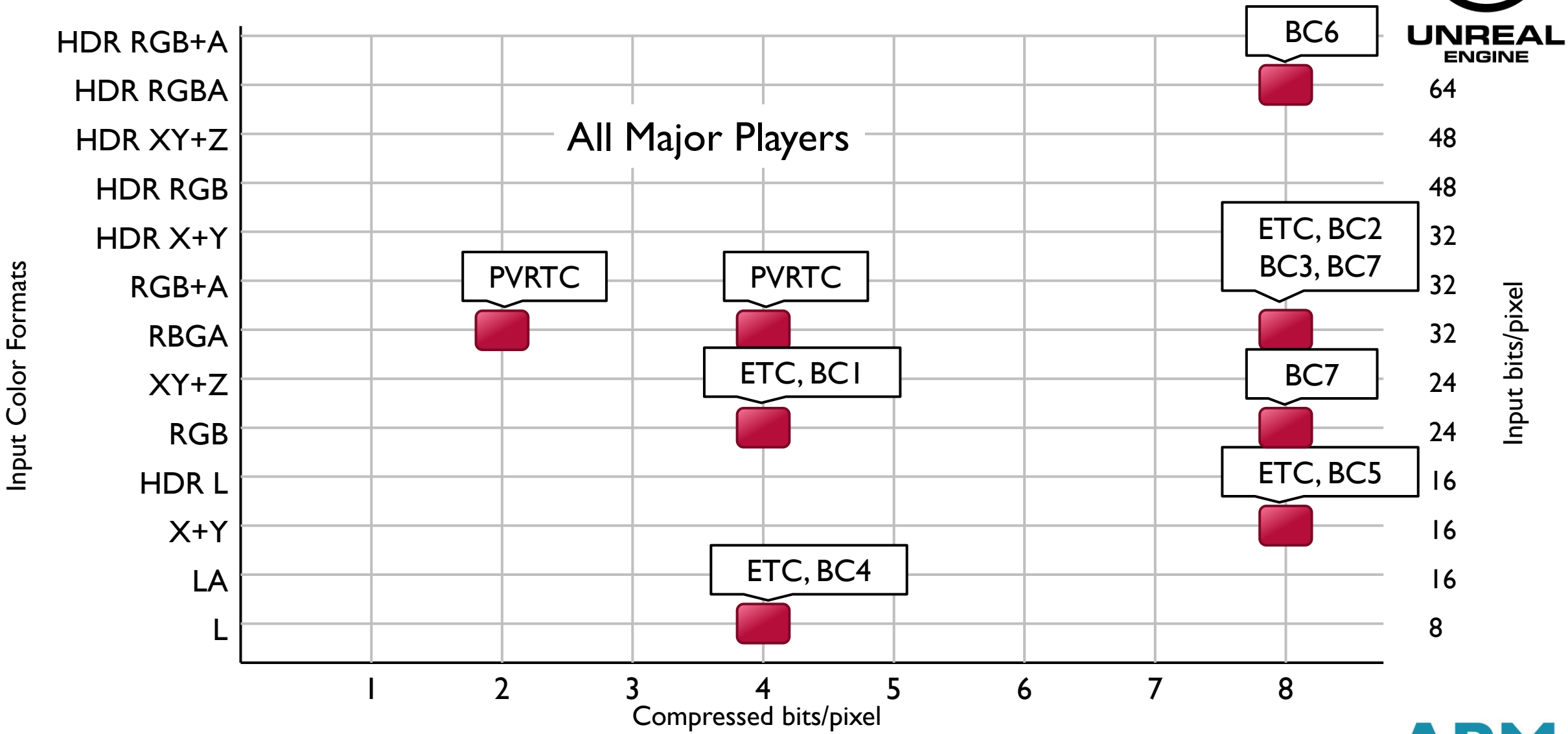
Niklas "Smedis" Smedberg
Senior Engine Programmer, Epic Games
Brad Grantham
Principal Software Engineer, ARM

Compress, Compress, Compress!

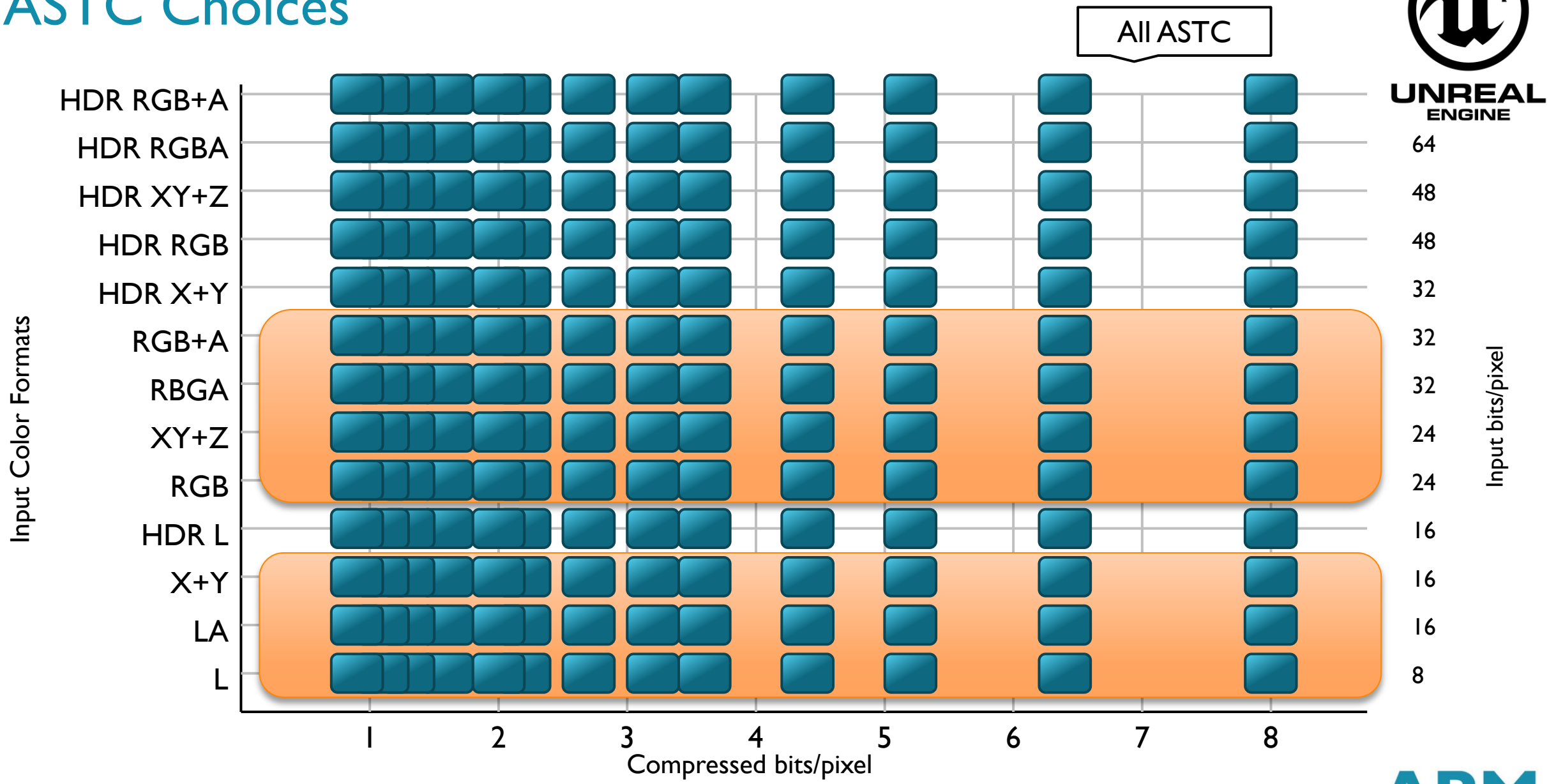
- ASTC = Adaptive Scalable Texture Compression
  - Texture compression standard developed by ARM, adopted by Khronos
    - KHR_texture_compression_astc_ldr for OpenGL ES and Open GL®
  - Increased quality and fidelity at low bit-rates
  - Expansive range of input formats offers complete flexibility
    - Choice of base format, 2D and 3D plus addition of HDR formats



Compression in the Pre-ASTC World



ASTC Choices

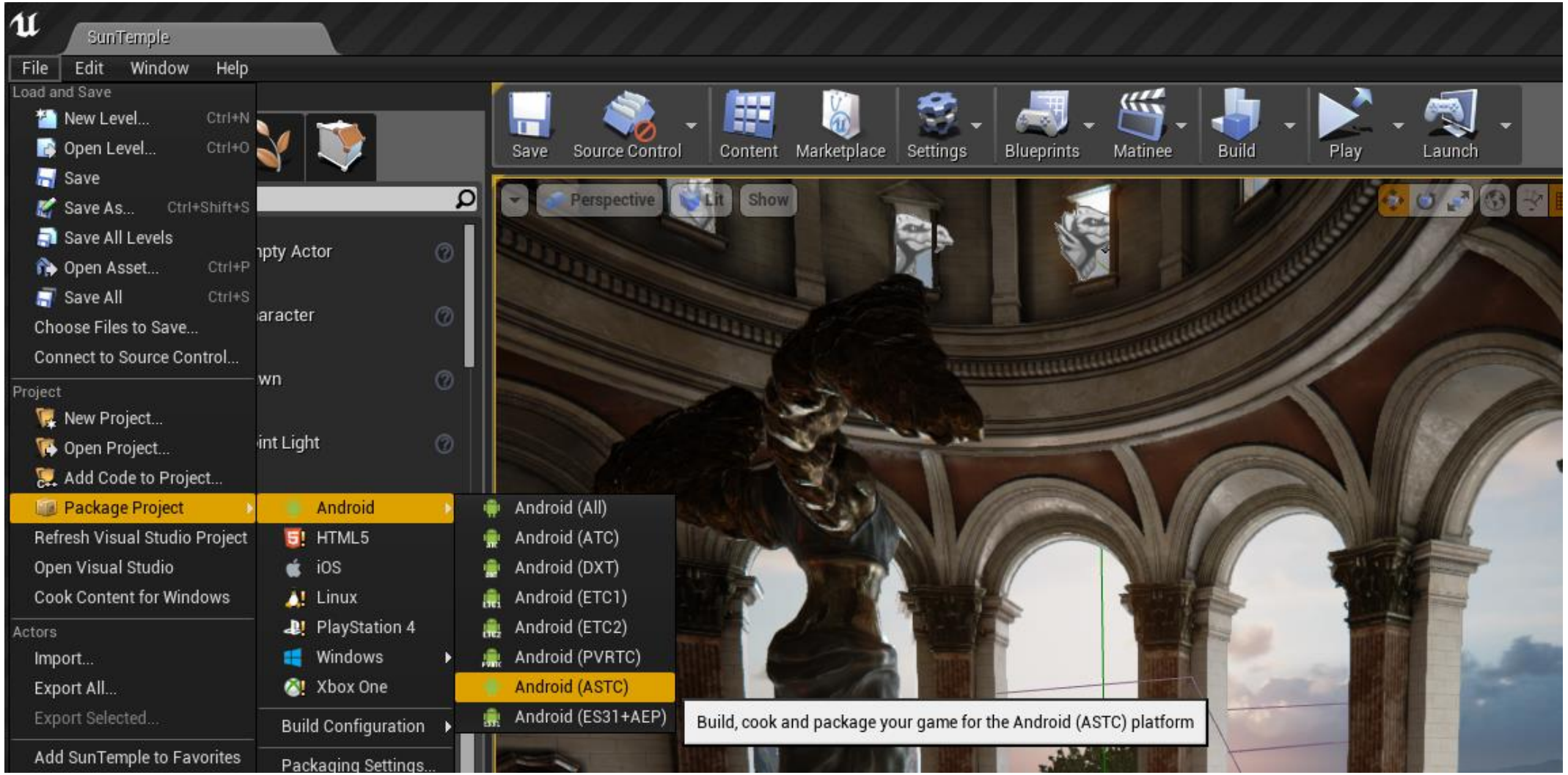


ASTC for Mobile Games

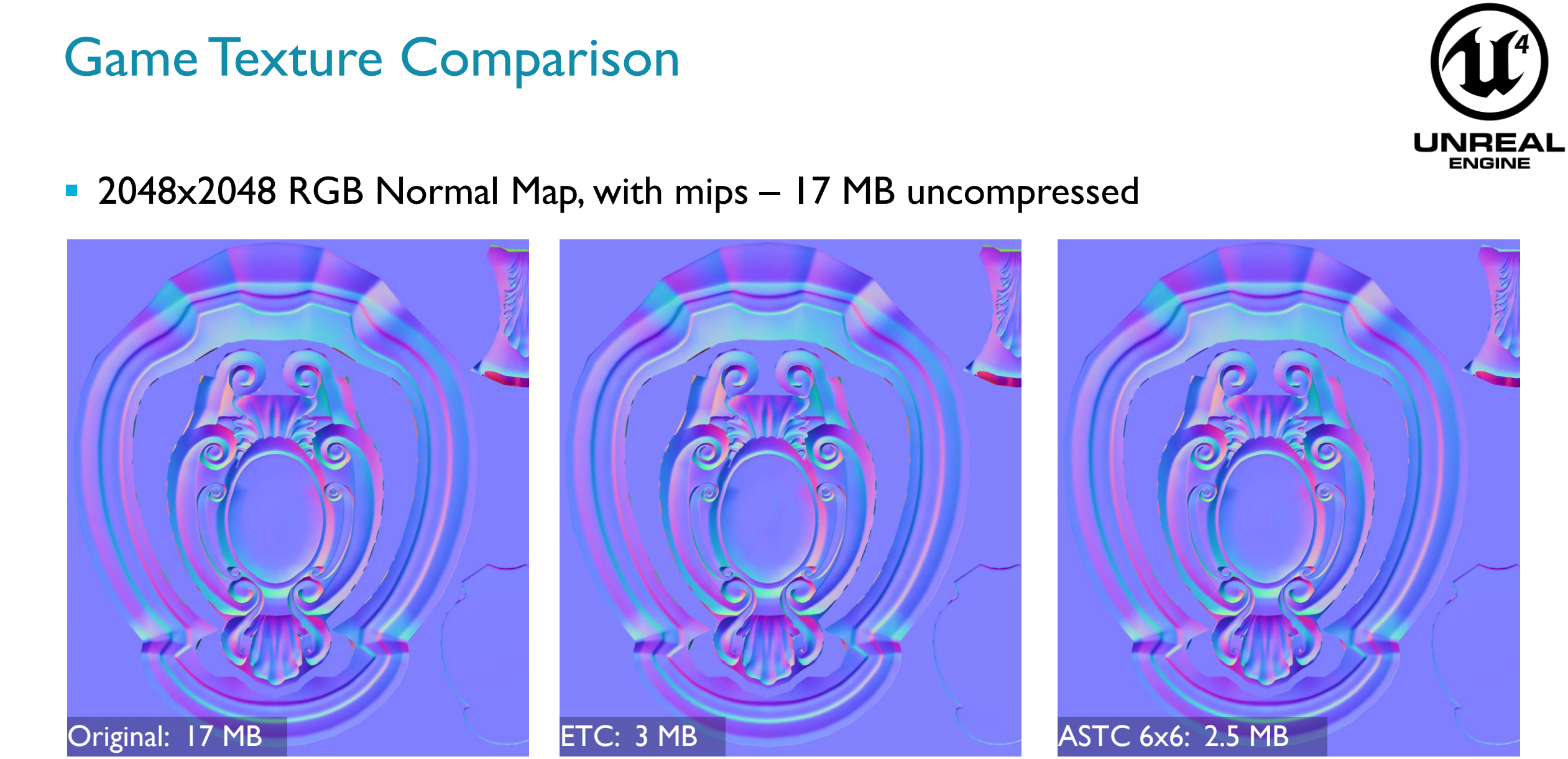
- ASTC is widely supported by all major hardware vendors
  - It's free to use
- Finally a good texture format that can work everywhere!
  - Avoids separate SKUs per hardware manufacturer: PVRTC, ATC, DXT, …
  - <supports-gl-texture android:name="GL_AMD_compressed_ATC_texture" />
- Support for ASTC is also required by Google's Android Extension Pack
  - GL_ANDROID_extension_pack_es31a

ARM

ASTC Support in Unreal Engine 4



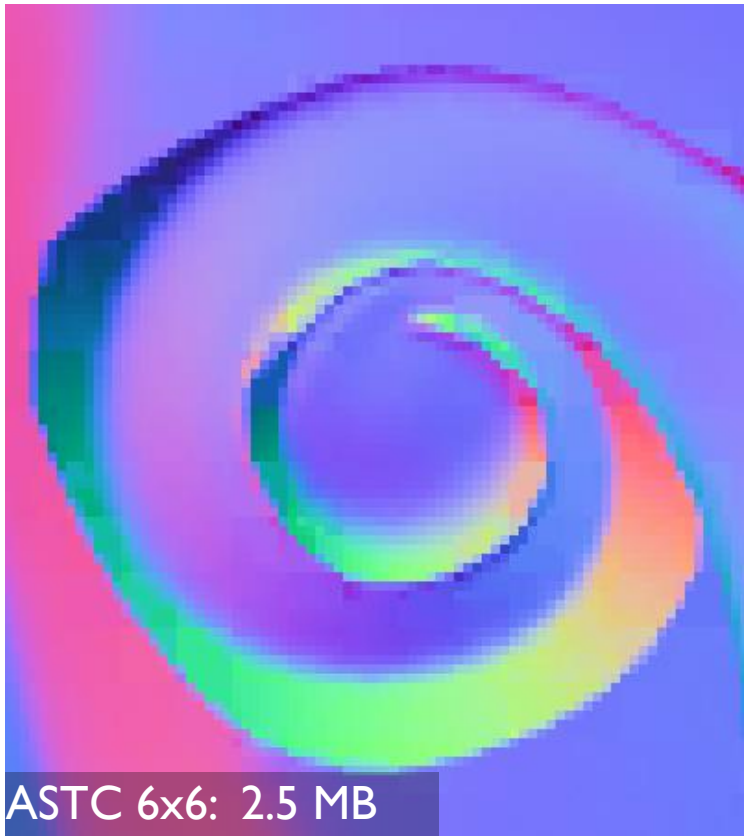
Game Texture Comparison



- 2048x2048 RGB Normal Map, with mips – 17 MB uncompressed

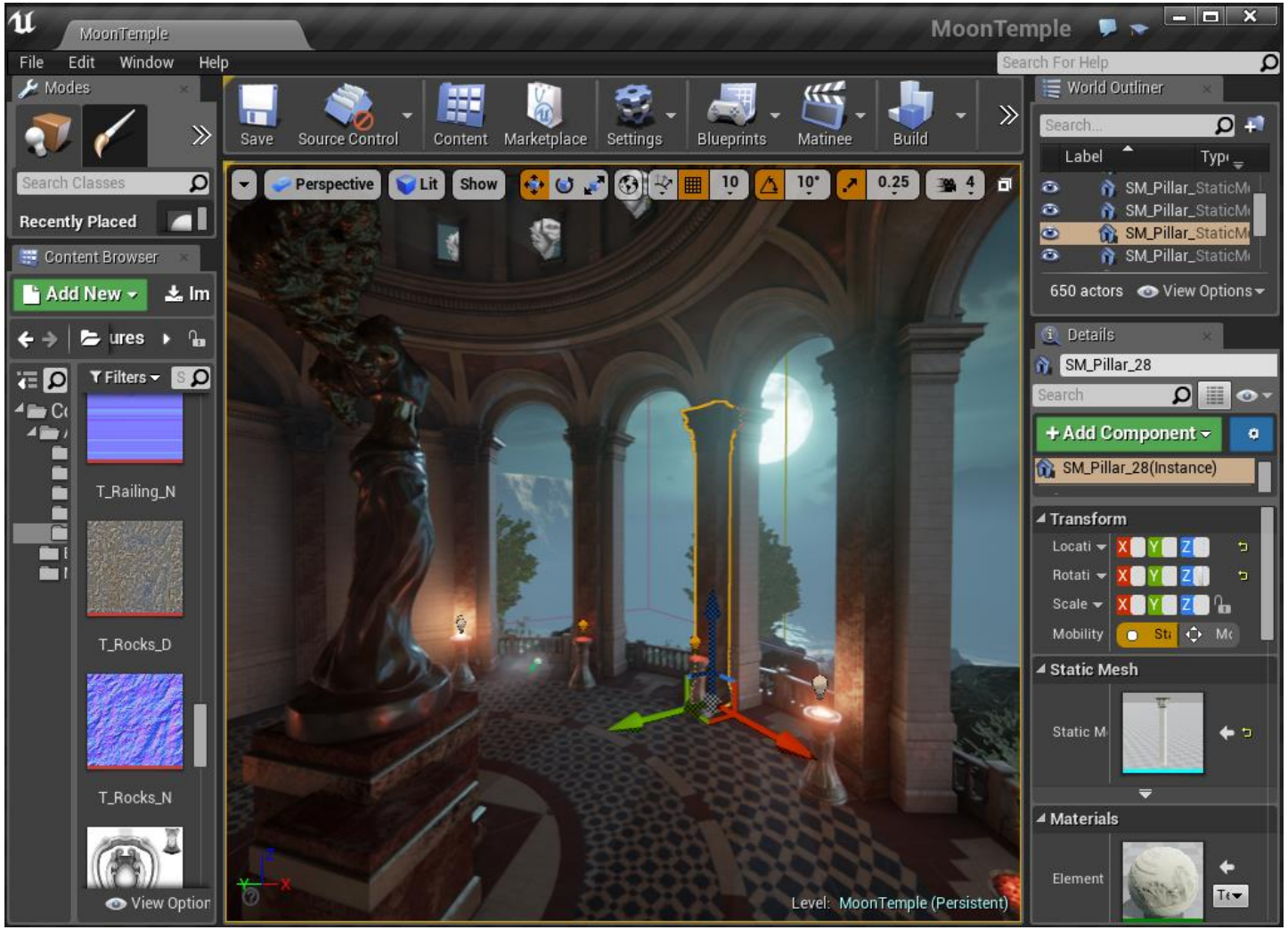
Game Texture Comparison

- Same texture – zoomed in for Truth



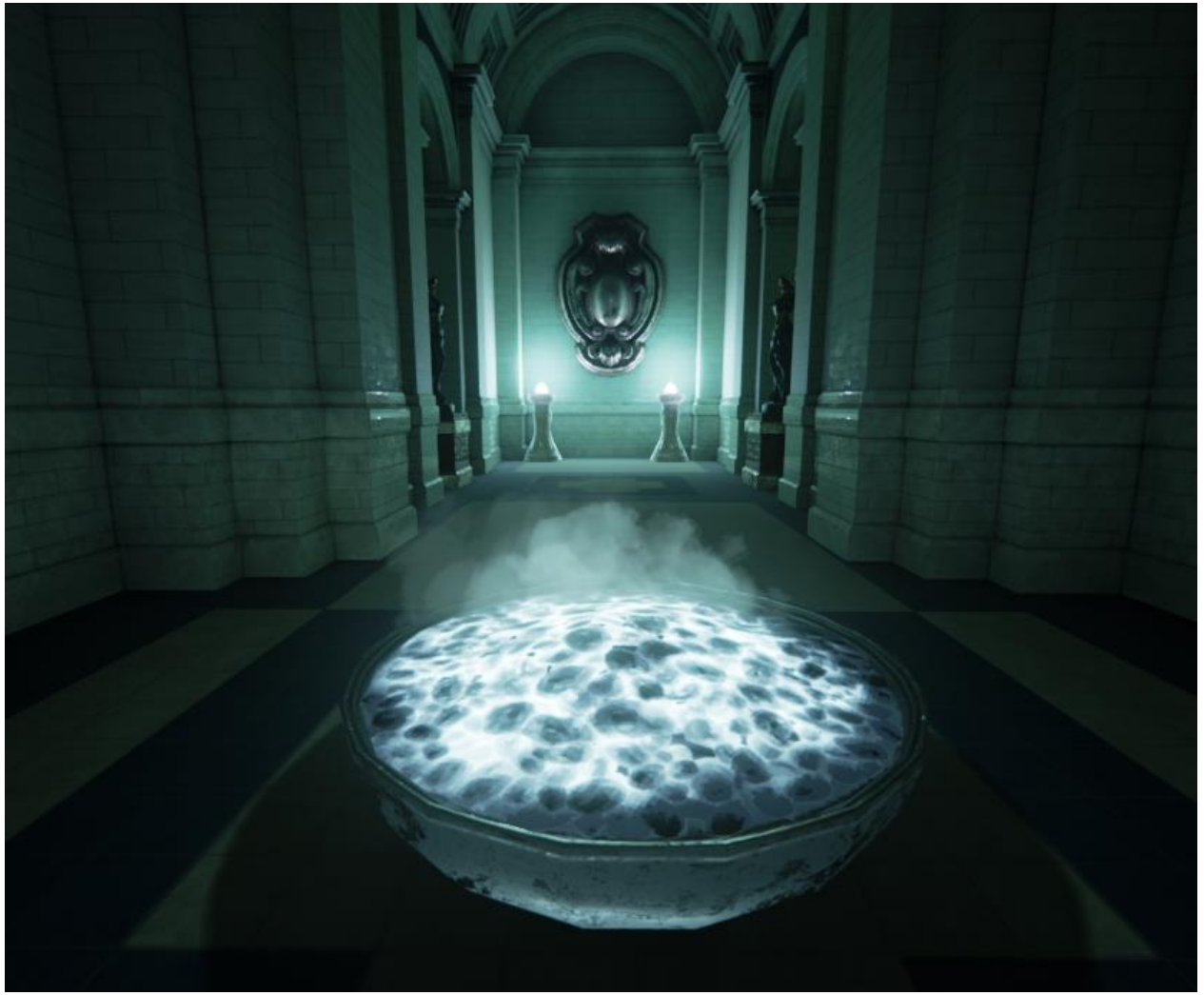
Unreal Engine 4 Demo: Moon Temple

- Made specifically for ARM
- Unreal Engine 4
- Goals:
  - 64-bit Android
  - ASTC
  - PLS



Unreal Engine 4 – Pixel Local Storage

- Read & write custom data for each pixel
- E.g. Depth
- Blend particles softly against the background



Unreal Engine 4 – Pixel Local Storage

Moon Temple Demo

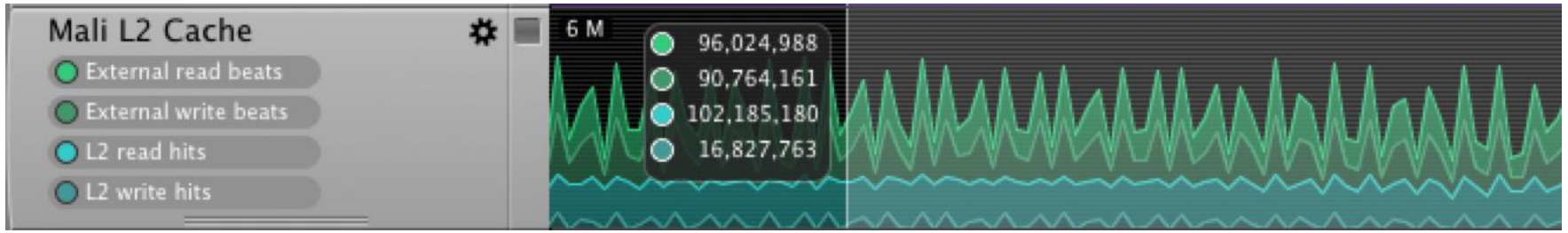
Enabling 64-bit Android in Unreal Engine 4

- Android NDK r10c
  - 64-bit AArch64 compilers
- Android SDK 21
  - Required for Lollipop, 64-bit
- UE4 Engine changes – collaboration between ARM and Epic Games
  - Patches submitted
  - Available in future release – packaging considerations to resolve
  - New Android platform "arm64", 64-bit libUE4.so
- Results: 8% Sun Temple FPS uplift just from compiling 64-bit

ARM

Measuring ASTC Benefit

- Streamline tool, part of ARM® Development Studio 5 (DS-5)
  - to know more https://ds.arm.com
- Capture CPU and GPU parameters during runtime for analysis
- ASTC requires less memory, so bandwidth use should drop
  - We should see that reflected in L2 cache external R+W beats
  - Example image from Streamline



Measuring ASTC Benefit

- Result of Streamline L2 counters:
- ETC2 over 30s: **1.29 GB/s**
- ASTC 6x6 over same 30s: **.98 GB/s**
- **24.4%** less bandwidth used per frame
- … And ASTC OBB is 12% smaller than ETC2 OBB (179MB versus 203MB)

ARM

Enlighten in Unreal Engine 4

Graham Hazel
Senior Product Manager

Geomerics
An ARM company

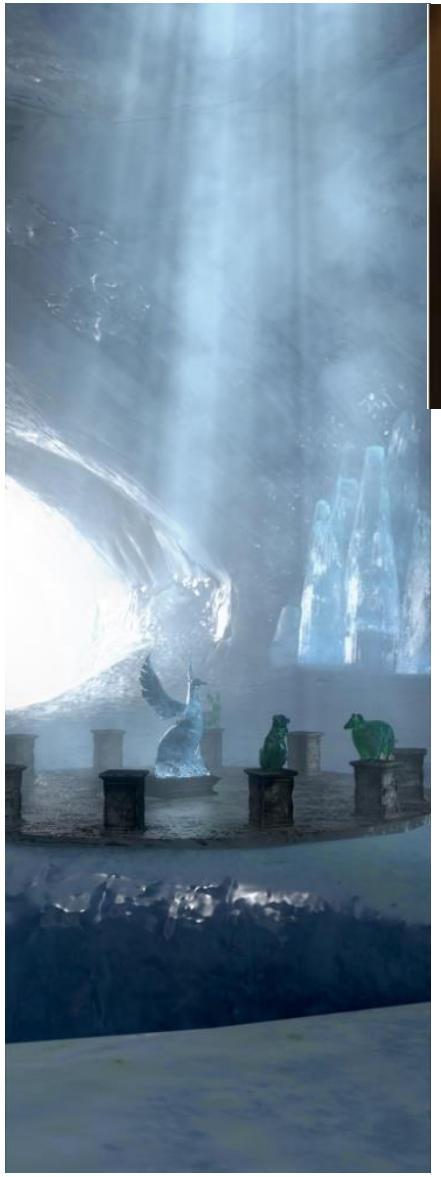
Enlighten in Unreal Engine 4

- Enlighten is global illumination middleware, available pre-integrated into UE4
- Runtime is lightweight and optimised for a wide range of platforms, including
  - Android 64-bit
  - iOS 64-bit
  - Windows PC
  - Mac OS X
  - PlayStation 4
  - Xbox One
- Find out more **Thursday 10AM, West Hall 3014**, and at the **ARM Booth 1624**

ARM

Enlighten in Unreal Engine 4

To Find Out More….



- ARM Booth #1624 on Expo Floor
  - Live demos
  - In-depth Q&A with ARM engineers
  - More tech talks at the ARM Lecture Theatre
    - **Epic Games: Live Session with Unreal Engine 4 for Mobile Devices**
    - **Geomerics Enlighten session**
    - **ARM tools Live Sessions**
- **http://malideveloper.arm.com/GDC2015**
  - Revisit this talk in PDF and video format post GDC
  - Download the tools and resources

More Talks from ARM at GDC 2015

Available post-show online at Mali Developer Center

- Unreal Engine 4 mobile graphics and the latest ARM CPU and GPU architecture - Weds 9:30AM; West Hall 3003

  This talk introduces the latest advances in features and benefits of the ARMv8-A and tile-based Mali GPU architectures on Unreal Engine 4, allowing mobile game developers to move to 64-bit's improved instruction set.

- Unleash the benefits of OpenGL ES 3.1 and Android Extension Pack (AEP) – Weds 2PM; West Hall 3003

  OpenGL ES 3.1 provides a rich set of tools for creating stunning images. This talk will cover best practices for using advanced features of OpenGL ES 3.1 on ARM Mali GPUs using recently developed examples from the Mali SDK.

- Making dreams come true – global illumination made easy – Thurs 10AM; West Hall 3014

  In this talk, we present an overview of the Enlighten feature set and show through workflow examples and gameplay demonstrations how it enables fast iteration and high visual quality on all gaming platforms.

- How to optimize your mobile game with ARM Tools and practical examples – Thurs 11:30AM; West Hall 3014

  This talk introduces you to the tools and skills needed to profile and debug your application by showing you optimization examples from popular game titles.

- Enhancing your Unity mobile game – Thurs 4PM; West Hall 3014

  Learn how to get the most out of Unity when developing under the unique challenges of mobile platforms.

Any Questions?

UNREAL ENGINE

Ask the best question and win a PiPO P4 tablet!

- Rockchip RK3288 processor
- ARM Cortex-A17 MP4 CPU
- ARM Mali-T760 MP4 GPU

ARM

Thank You