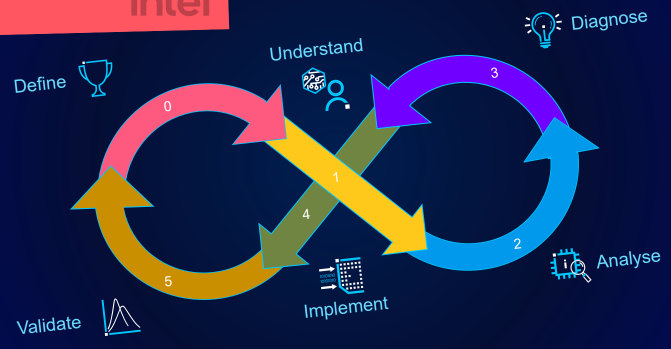
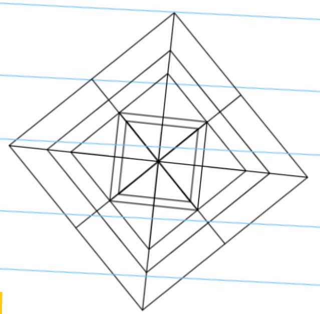
Graphics Programming Conference 2025, Breda

intel



The optimization process

Profiling in PIX



nce, November 18-20, Breda 4 2025

intel

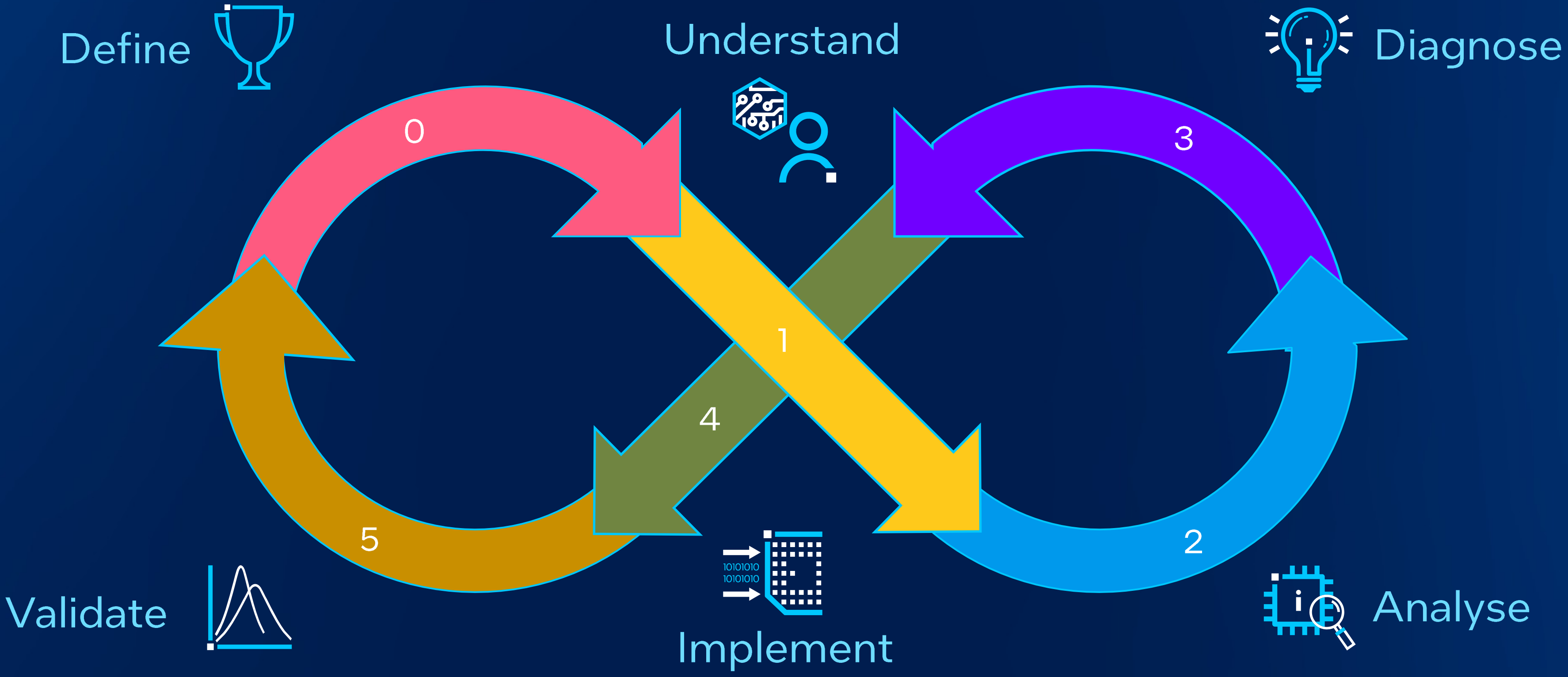
Steven Tovey
Intel

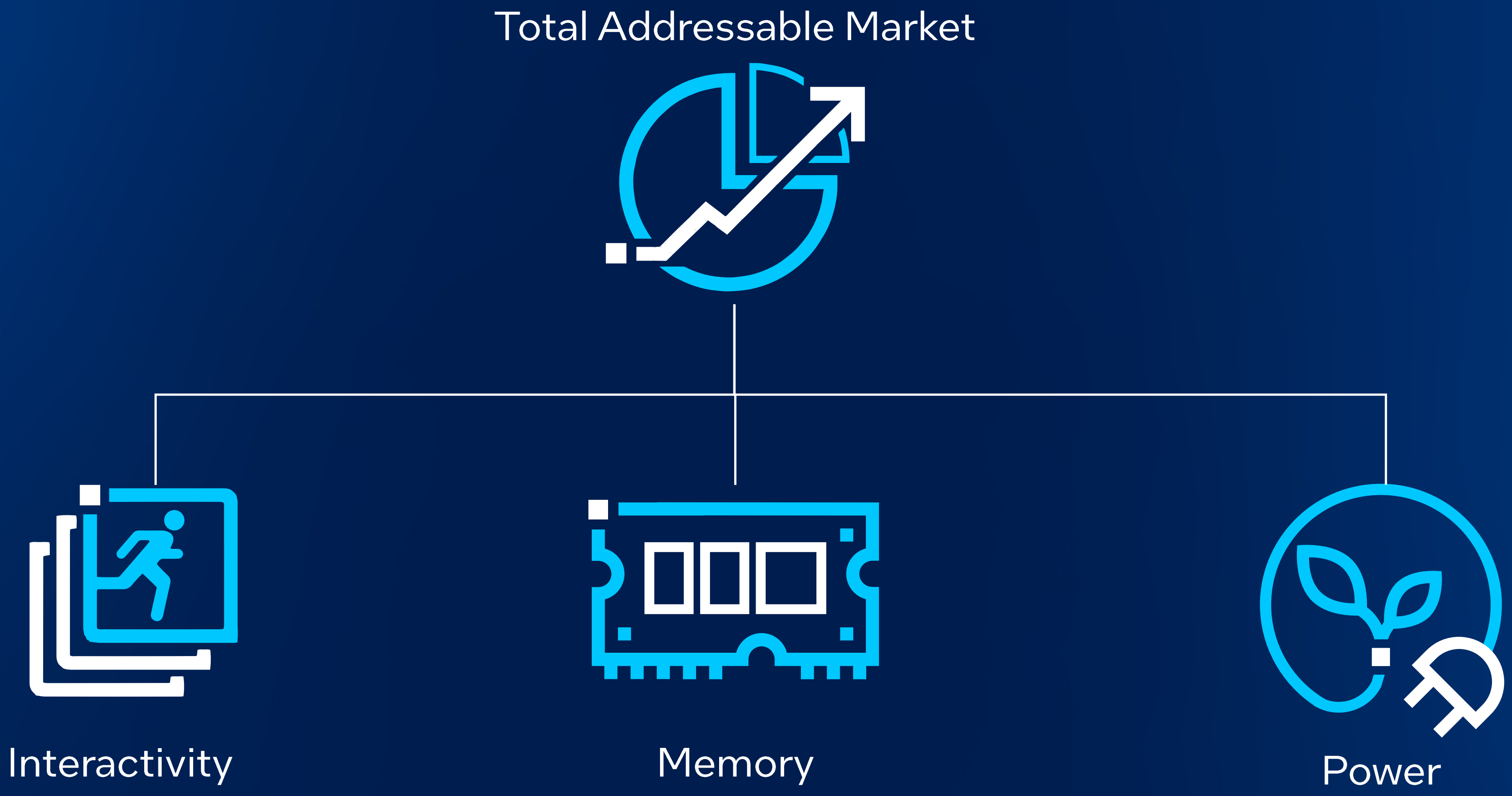
intel

intel

Intel
+
PIX

intel


Define
Understand
Diagnose
0
3
1
4
5
2
Validate
Implement
Analyse
10101010
10101010

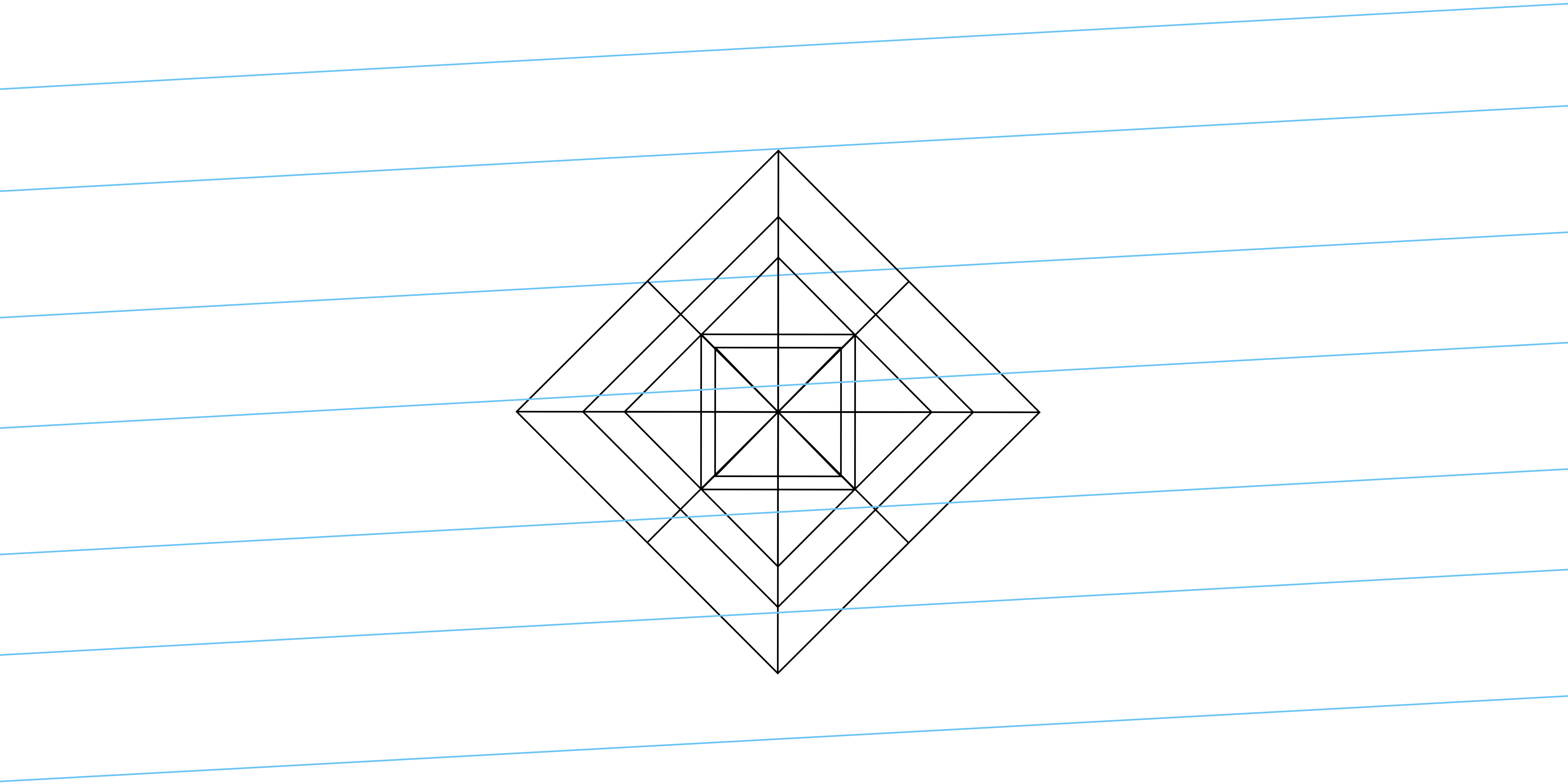


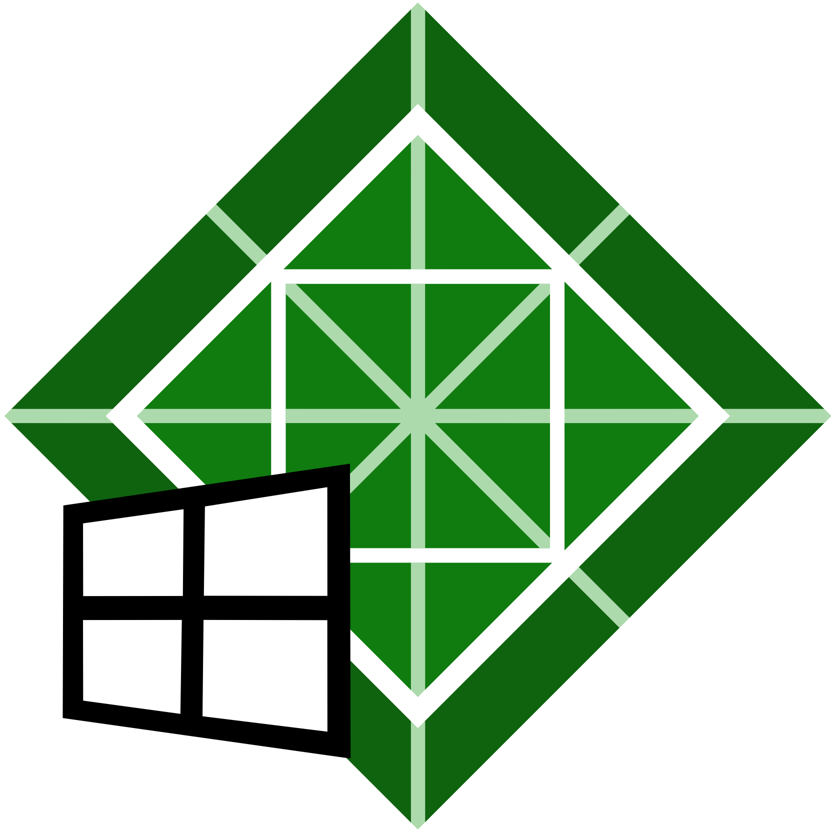
Total Addressable Market

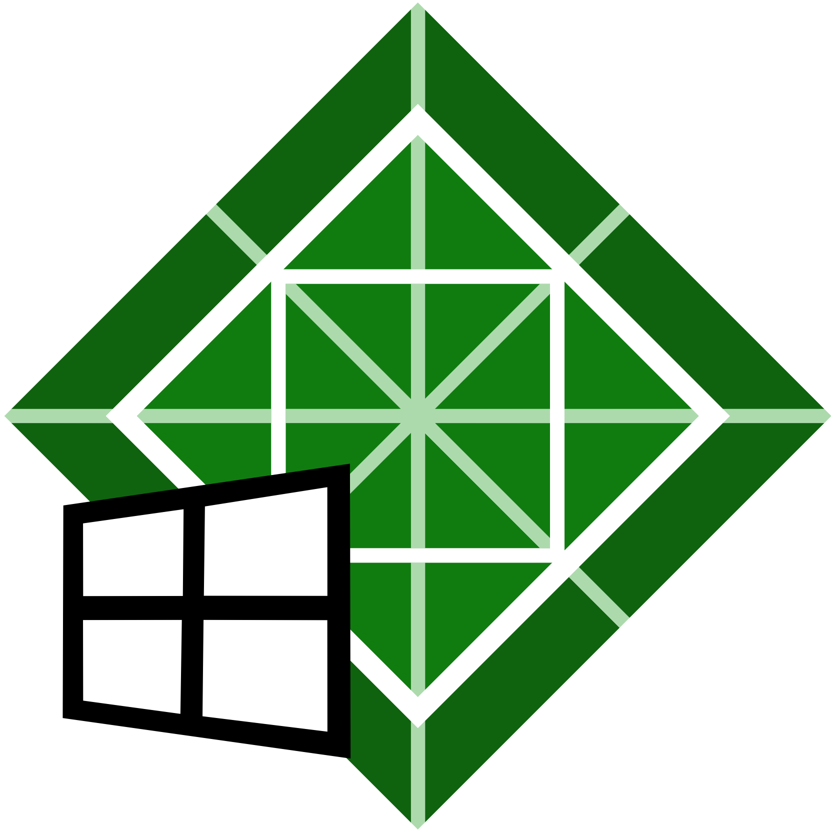
Interactivity

Memory

Power







First class metrics experience

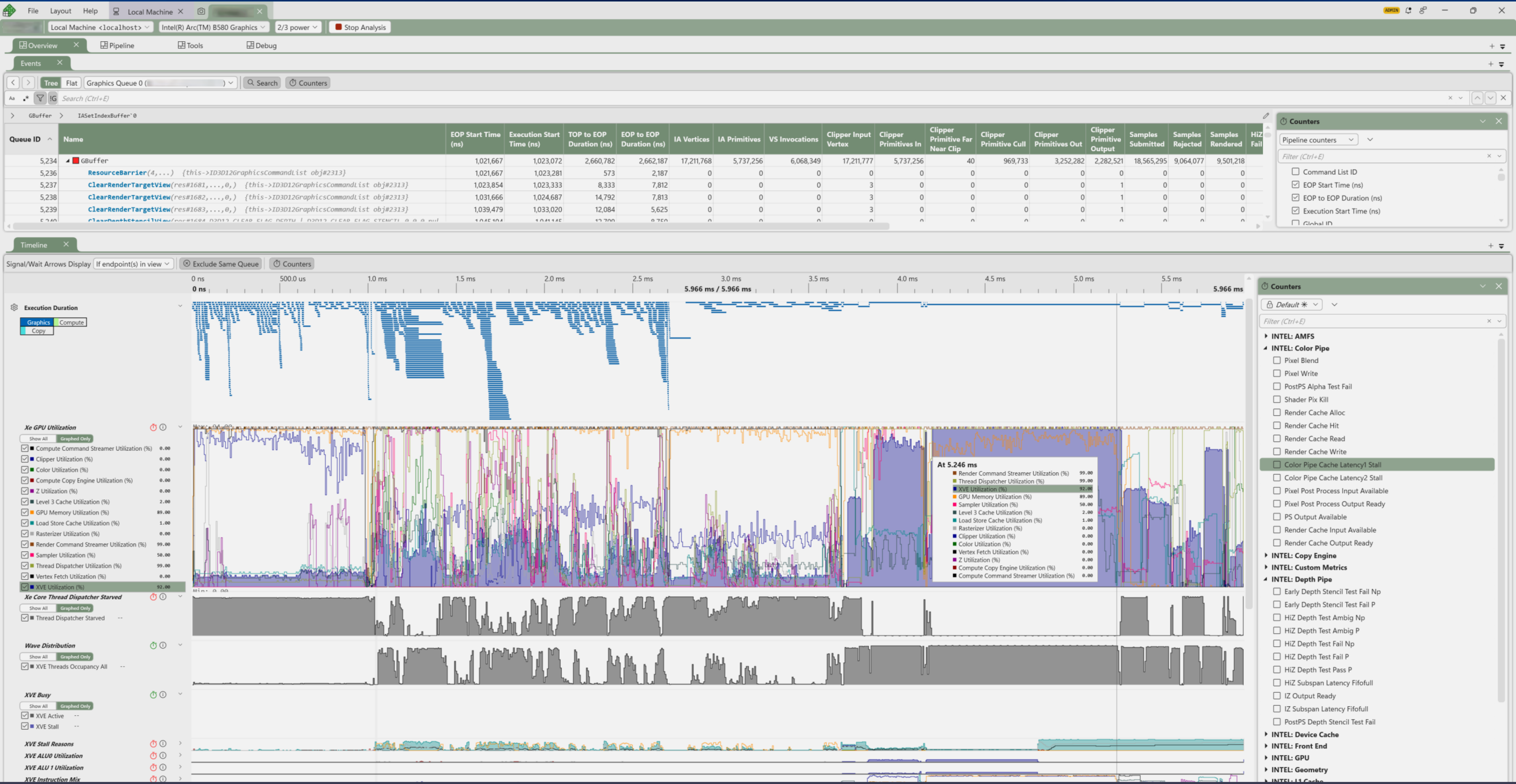
Stabilize, consolidate, and organize HW & derived metrics for ease of profiling.

In-depth shader profiling

XVE stall sampling,
with per-instruction visualisation
of stall reasons.

Crash debugging

Make it easier to diagnose root cause of GPU crashes and hangs.



Queue ID	Name	EOP Start Time (ns)	Execution Start Time (ns)	TOP to EOP Duration (ns)	EOP to EOP Duration (ns)	IA Vertices	IA Primitives	VS Invocations	Clipper Input Vertex	Clipper Primitives In	Clipper Primitive Far Near Clip	Clipper Primitive Cull	Clipper Primitives Out	Clipper Primitive Output	Samples Submitted	Samples Rejected	Samples Rendered
5,234	🟥 GBuffer	1,021,667	1,023,072	2,660,782	2,662,187	17,211,768	5,737,256	6,068,349	17,211,777	5,737,256	40	969,733	3,252,282	2,282,521	18,565,295	9,064,077	9,501,218
5,236	`ResourceBarrier(4,...) {this->ID3D12GraphicsCommandList obj#2313}`	1,021,667	1,023,281	573	2,187	0	0	0	0	0	0	0	0	0	0	0	0
5,237	`ClearRenderTargetView(res#1681,...,0,) {this->ID3D12GraphicsCommandList obj#2313}`	1,023,854	1,023,333	8,333	7,812	0	0	0	3	0	0	0	0	1	0	0	0
5,238	`ClearRenderTargetView(res#1682,...,0,) {this->ID3D12GraphicsCommandList obj#2313}`	1,031,666	1,024,687	14,792	7,813	0	0	0	3	0	0	0	0	1	0	0	0
5,239	`ClearRenderTargetView(res#1683,...,0,) {this->ID3D12GraphicsCommandList obj#2313}`	1,039,479	1,033,020	12,084	5,625	0	0	0	3	0	0	0	0	1	0	0	0

Counters

Pipeline counters

Filter (Ctrl+E)

☐ Command List ID
☑ EOP Start Time (ns)
☑ EOP to EOP Duration (ns)
☑ Execution Start Time (ns)
☐ Global ID

Xe GPU Utilization

	Counter	Value
☑	Compute Command Streamer Utilization (%)	0.00
☑	Clipper Utilization (%)	0.00
☑	Color Utilization (%)	0.00
☑	Compute Copy Engine Utilization (%)	0.00
☑	Z Utilization (%)	0.00
☑	Level 3 Cache Utilization (%)	2.00
☑	GPU Memory Utilization (%)	89.00
☑	Load Store Cache Utilization (%)	1.00
☑	Rasterizer Utilization (%)	0.00
☑	Render Command Streamer Utilization (%)	99.00
☑	Sampler Utilization (%)	50.00
☑	Thread Dispatcher Utilization (%)	99.00
☑	Vertex Fetch Utilization (%)	0.00
☑	XVE Utilization (%)	92.00

Xe Core Thread Dispatcher Starved

☑ Thread Dispatcher Starved --

Wave Distribution

☑ XVE Threads Occupancy All --

XVE Busy

☑ XVE Active --
☑ XVE Stall --

At 5.246 ms

Counter	Value
Render Command Streamer Utilization (%)	99.00
Thread Dispatcher Utilization (%)	99.00
XVE Utilization (%)	92.00
GPU Memory Utilization (%)	89.00
Sampler Utilization (%)	50.00
Level 3 Cache Utilization (%)	2.00
Load Store Cache Utilization (%)	1.00
Rasterizer Utilization (%)	0.00
Clipper Utilization (%)	0.00
Color Utilization (%)	0.00
Vertex Fetch Utilization (%)	0.00
Z Utilization (%)	0.00
Compute Copy Engine Utilization (%)	0.00
Compute Command Streamer Utilization (%)	0.00

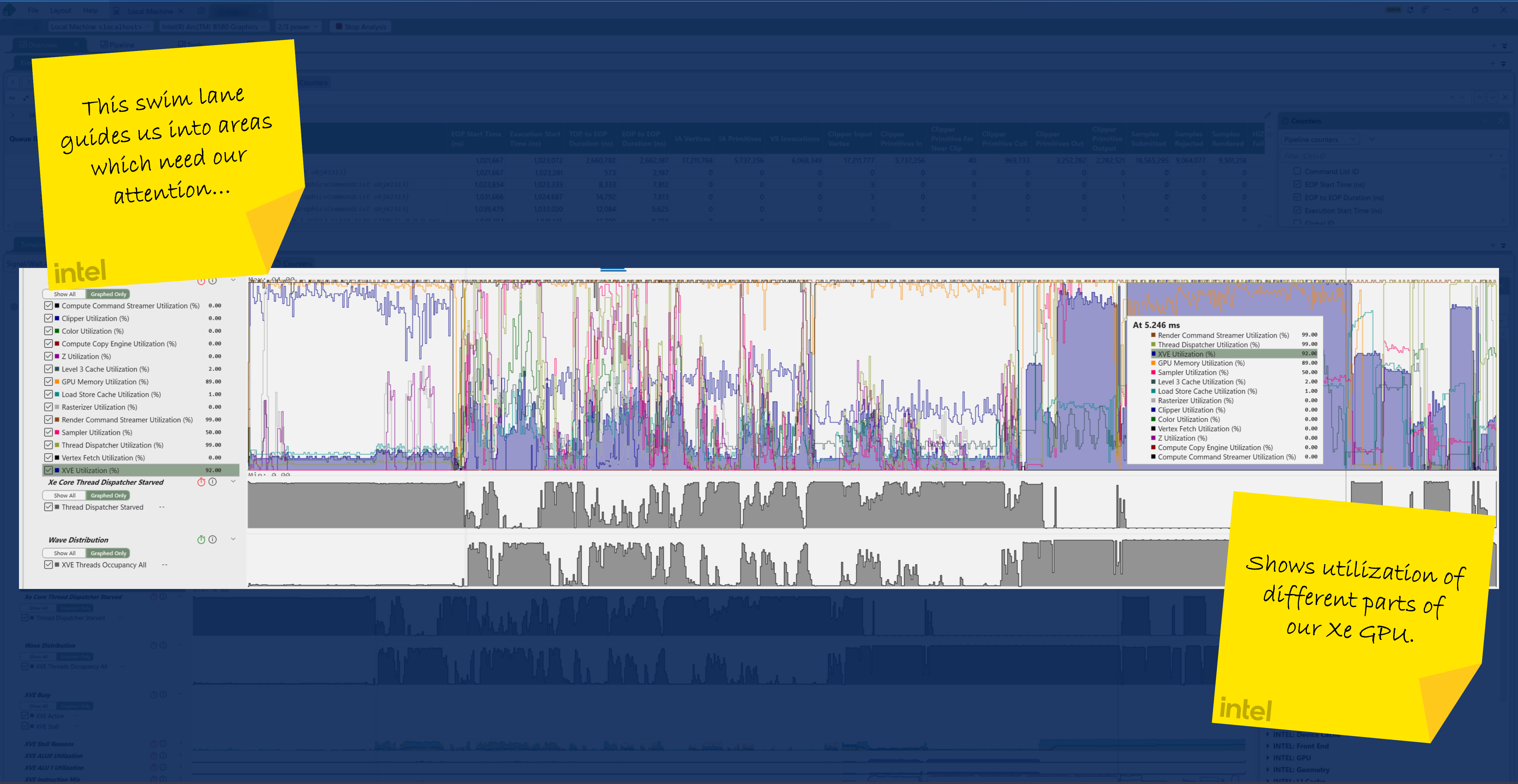
Counters

Default

Filter (Ctrl+E)

- INTEL: AMFS
- INTEL: Color Pipe
  - ☐ Pixel Blend
  - ☐ Pixel Write
  - ☐ PostPS Alpha Test Fail
  - ☐ Shader Pix Kill
  - ☐ Render Cache Alloc
  - ☐ Render Cache Hit
  - ☐ Render Cache Read
  - ☐ Render Cache Write
  - ☐ Color Pipe Cache Latency1 Stall
  - ☐ Color Pipe Cache Latency2 Stall
  - ☐ Pixel Post Process Input Available
  - ☐ Pixel Post Process Output Ready
  - ☐ PS Output Available
  - ☐ Render Cache Input Available
  - ☐ Render Cache Output Ready
- INTEL: Copy Engine
- INTEL: Custom Metrics
- INTEL: Depth Pipe
  - ☐ Early Depth Stencil Test Fail Np
  - ☐ Early Depth Stencil Test Fail P
  - ☐ HiZ Depth Test Ambig Np
  - ☐ HiZ Depth Test Ambig P
  - ☐ HiZ Depth Test Fail Np
  - ☐ HiZ Depth Test Fail P
  - ☐ HiZ Depth Test Pass P
  - ☐ HiZ Subspan Latency Fifofull
  - ☐ IZ Output Ready
  - ☐ IZ Subspan Latency Fifofull
  - ☐ PostPS Depth Stencil Test Fail
- INTEL: Device Cache
- INTEL: Front End
- INTEL: GPU
- INTEL: Geometry

This swim lane guides us into areas which need our attention...



☑ Compute Command Streamer Utilization (%)	0.00
☑ Clipper Utilization (%)	0.00
☑ Color Utilization (%)	0.00
☑ Compute Copy Engine Utilization (%)	0.00
☑ Z Utilization (%)	0.00
☑ Level 3 Cache Utilization (%)	2.00
☑ GPU Memory Utilization (%)	89.00
☑ Load Store Cache Utilization (%)	1.00
☑ Rasterizer Utilization (%)	0.00
☑ Render Command Streamer Utilization (%)	99.00
☑ Sampler Utilization (%)	50.00
☑ Thread Dispatcher Utilization (%)	99.00
☑ Vertex Fetch Utilization (%)	0.00
☑ XVE Utilization (%)	92.00

Xe Core Thread Dispatcher Starved

☑ Thread Dispatcher Starved	--

Wave Distribution

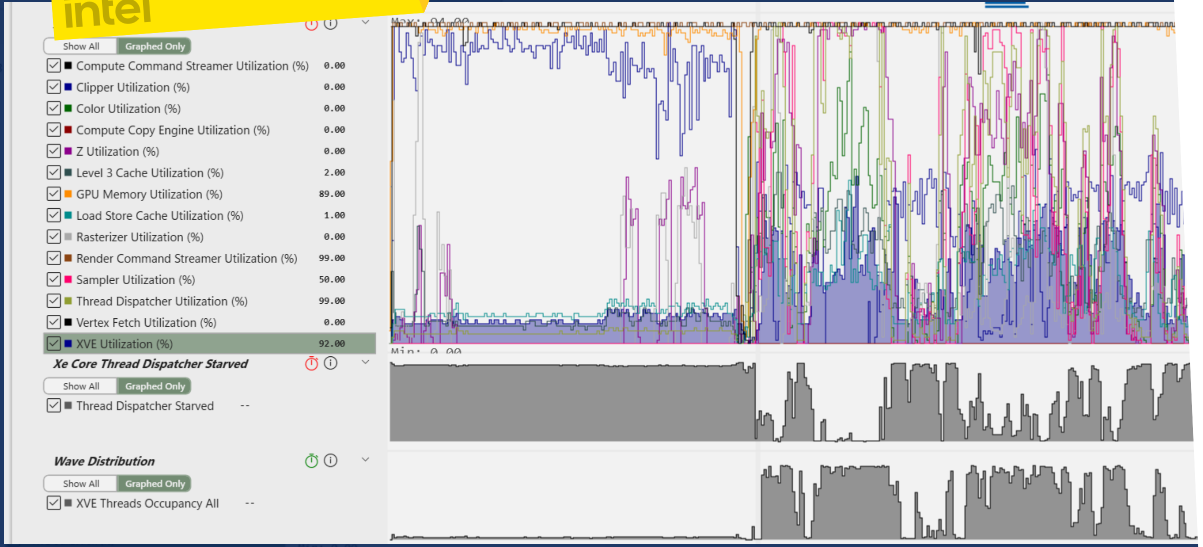
☑ XVE Threads Occupancy All	--

At 5.246 ms

Render Command Streamer Utilization (%)	99.00
Thread Dispatcher Utilization (%)	99.00
XVE Utilization (%)	92.00
GPU Memory Utilization (%)	89.00
Sampler Utilization (%)	50.00
Level 3 Cache Utilization (%)	2.00
Load Store Cache Utilization (%)	1.00
Rasterizer Utilization (%)	0.00
Clipper Utilization (%)	0.00
Color Utilization (%)	0.00
Vertex Fetch Utilization (%)	0.00
Z Utilization (%)	0.00
Compute Copy Engine Utilization (%)	0.00
Compute Command Streamer Utilization (%)	0.00

Shows utilization of different parts of our Xe GPU.

This swim lane guides us into areas which need our attention...


Show All | Graphed Only
☑ Compute Command Streamer Utilization (%) 0.00
☑ Clipper Utilization (%) 0.00
☑ Color Utilization (%) 0.00
☑ Compute Copy Engine Utilization (%) 0.00
☑ Z Utilization (%) 0.00
☑ Level 3 Cache Utilization (%) 2.00
☑ GPU Memory Utilization (%) 89.00
☑ Load Store Cache Utilization (%) 1.00
☑ Rasterizer Utilization (%) 0.00
☑ Render Command Streamer Utilization (%) 99.00
☑ Sampler Utilization (%) 50.00
☑ Thread Dispatcher Utilization (%) 99.00
☑ Vertex Fetch Utilization (%) 0.00
☑ XVE Utilization (%) 92.00

Xe Core Thread Dispatcher Starved

Show All | Graphed Only
☑ Thread Dispatcher Starved --

Wave Distribution

Show All | Graphed Only
☑ XVE Threads Occupancy All --

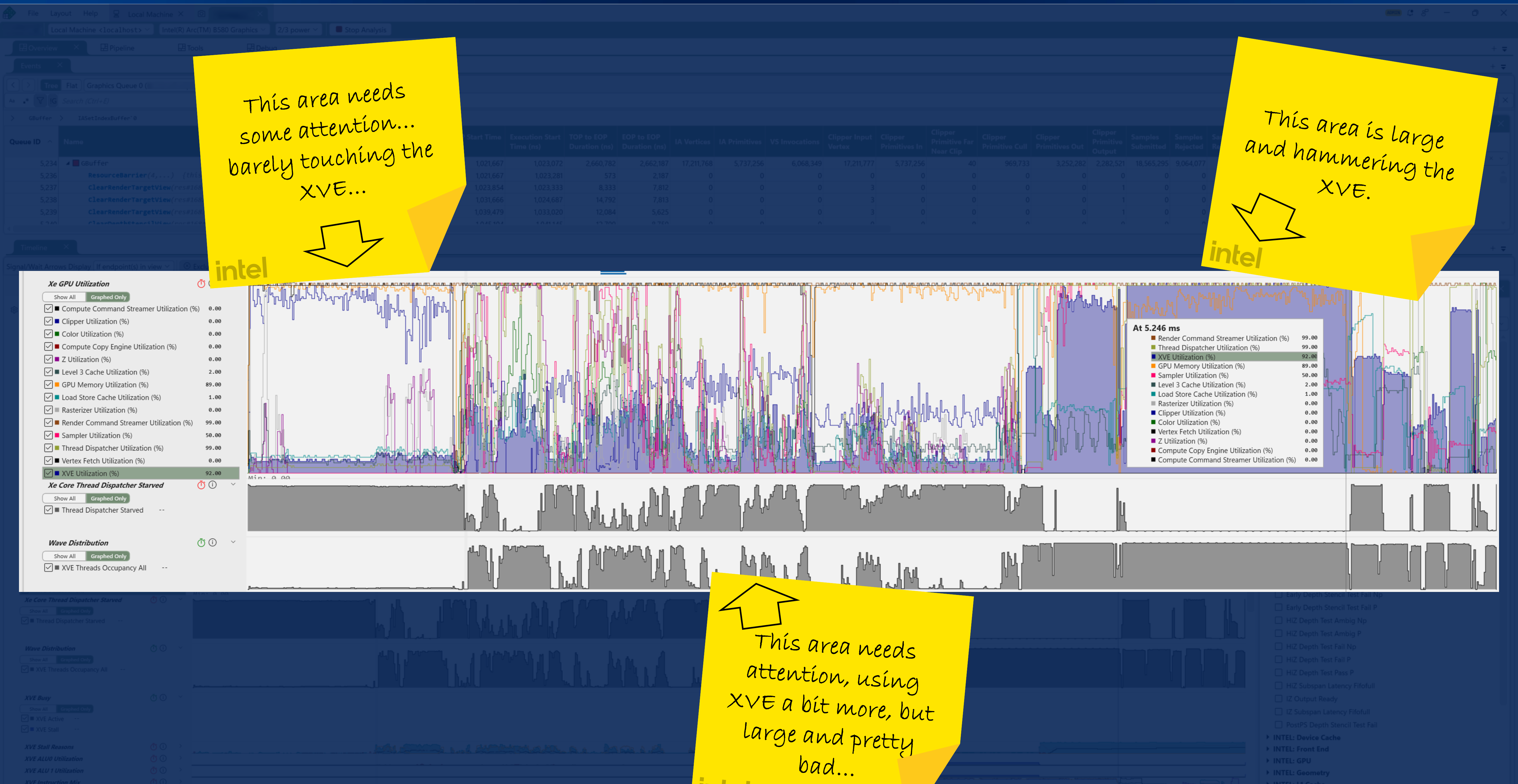
Choosing a focus area...

- Is it representative of my application?
- Is it taking a chunk of frame?

vs utilization of
ferent parts of
ur Xe GPU.

This area needs some attention... barely touching the XVE...

This area is large and hammering the XVE.



Xe GPU Utilization

Show All | Graphed Only

- ☑ Compute Command Streamer Utilization (%) 0.00
- ☑ Clipper Utilization (%) 0.00
- ☑ Color Utilization (%) 0.00
- ☑ Compute Copy Engine Utilization (%) 0.00
- ☑ Z Utilization (%) 0.00
- ☑ Level 3 Cache Utilization (%) 2.00
- ☑ GPU Memory Utilization (%) 89.00
- ☑ Load Store Cache Utilization (%) 1.00
- ☑ Rasterizer Utilization (%) 0.00
- ☑ Render Command Streamer Utilization (%) 99.00
- ☑ Sampler Utilization (%) 50.00
- ☑ Thread Dispatcher Utilization (%) 99.00
- ☑ Vertex Fetch Utilization (%) 0.00
- ☑ XVE Utilization (%) 92.00

Xe Core Thread Dispatcher Starved

Show All | Graphed Only

- ☑ Thread Dispatcher Starved --

Wave Distribution

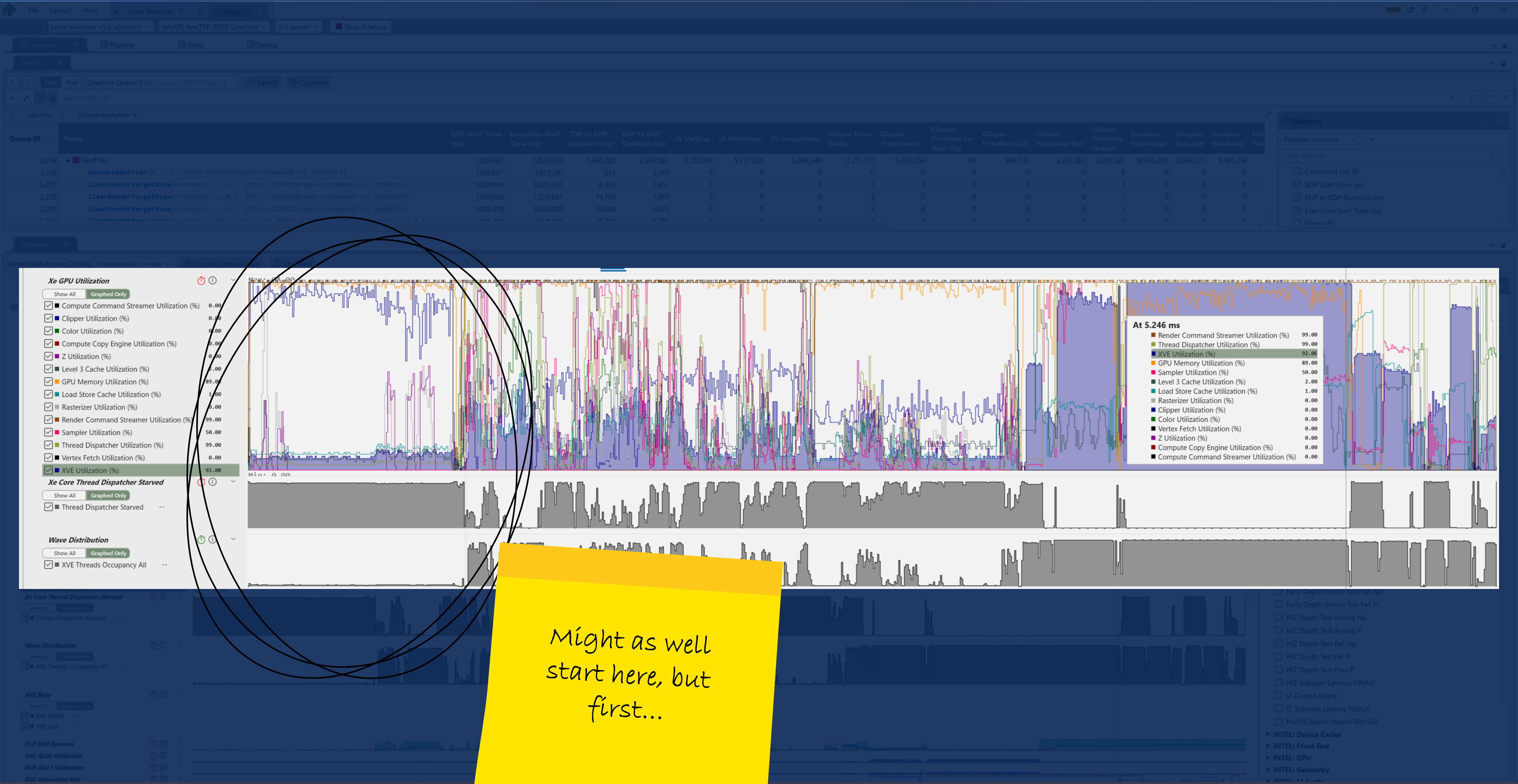
Show All | Graphed Only

- ☑ XVE Threads Occupancy All --

At 5.246 ms

Render Command Streamer Utilization (%)	99.00
Thread Dispatcher Utilization (%)	99.00
XVE Utilization (%)	92.00
GPU Memory Utilization (%)	89.00
Sampler Utilization (%)	50.00
Level 3 Cache Utilization (%)	2.00
Load Store Cache Utilization (%)	1.00
Rasterizer Utilization (%)	0.00
Clipper Utilization (%)	0.00
Color Utilization (%)	0.00
Vertex Fetch Utilization (%)	0.00
Z Utilization (%)	0.00
Compute Copy Engine Utilization (%)	0.00
Compute Command Streamer Utilization (%)	0.00

This area needs attention, using XVE a bit more, but large and pretty bad...



Xe GPU Utilization

Show All | Graphed Only

☑ Compute Command Streamer Utilization (%) 0.00
☑ Clipper Utilization (%) 0.00
☑ Color Utilization (%) 0.00
☑ Compute Copy Engine Utilization (%) 0.00
☑ Z Utilization (%) 0.00
☑ Level 3 Cache Utilization (%) 2.00
☑ GPU Memory Utilization (%) 89.00
☑ Load Store Cache Utilization (%) 1.00
☑ Rasterizer Utilization (%) 0.00
☑ Render Command Streamer Utilization (%) 99.00
☑ Sampler Utilization (%) 50.00
☑ Thread Dispatcher Utilization (%) 99.00
☑ Vertex Fetch Utilization (%) 0.00
☑ XVE Utilization (%) 92.00

Xe Core Thread Dispatcher Starved

Show All | Graphed Only

☑ Thread Dispatcher Starved --

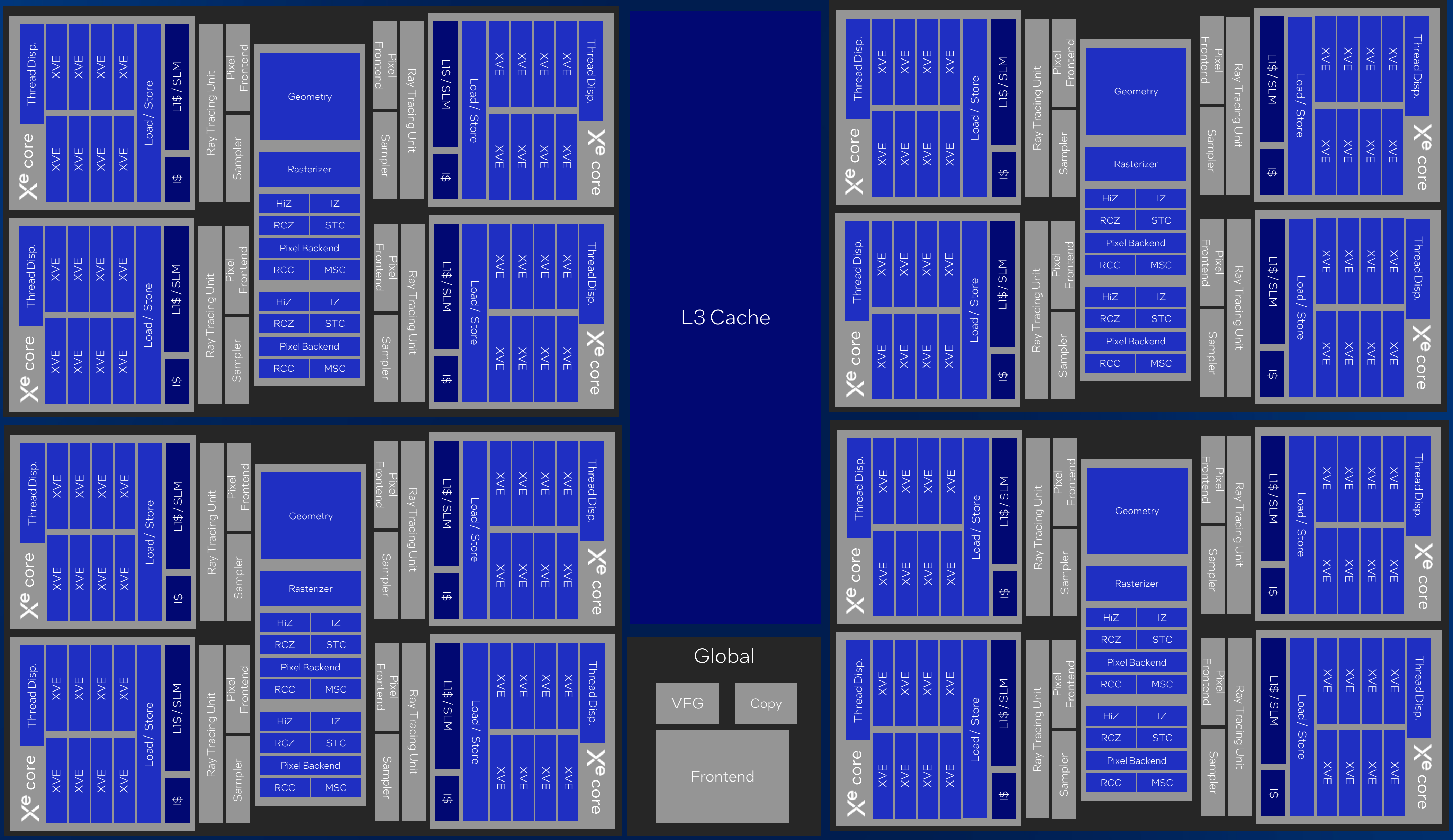
Wave Distribution

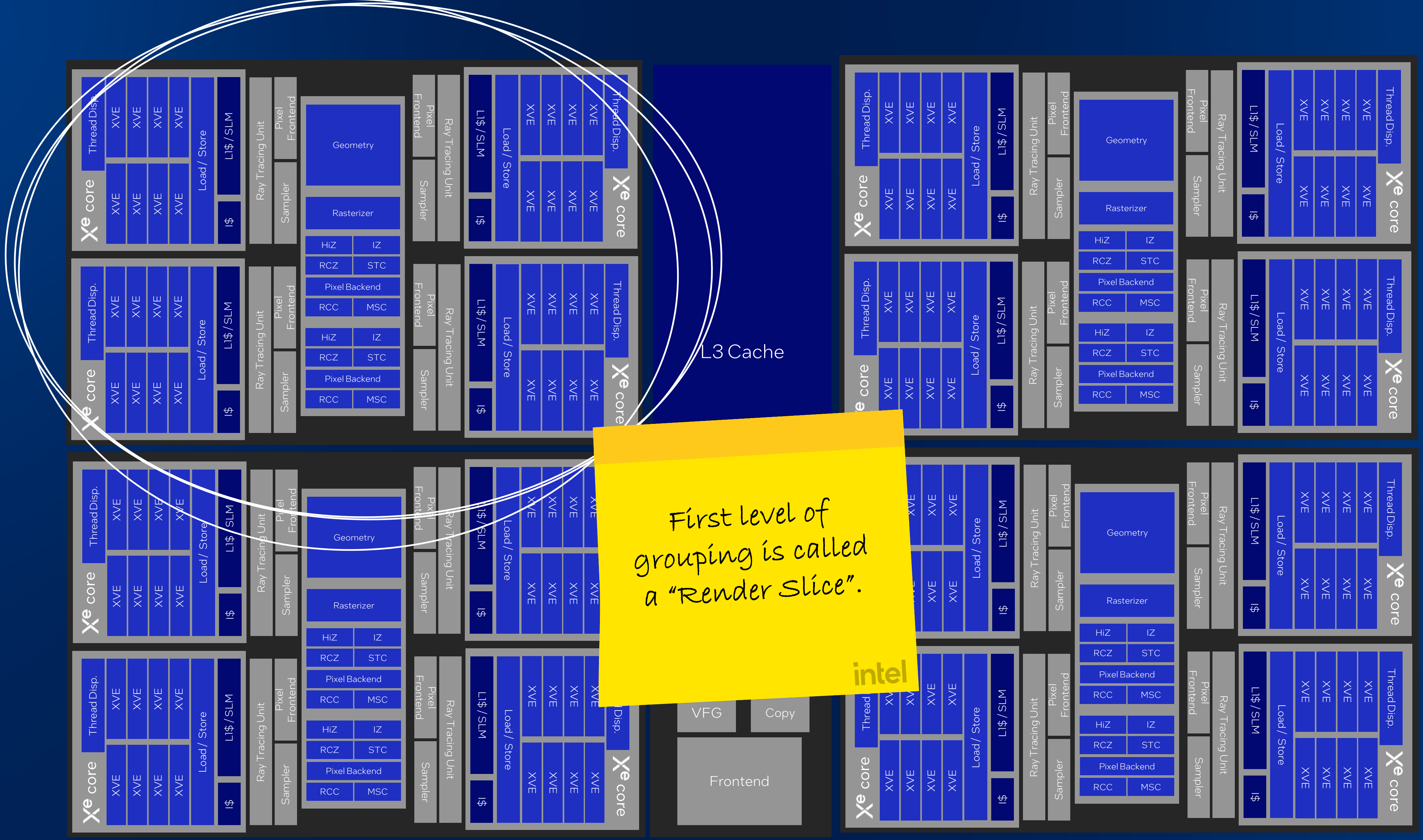
Show All | Graphed Only

☑ XVE Threads Occupancy All --

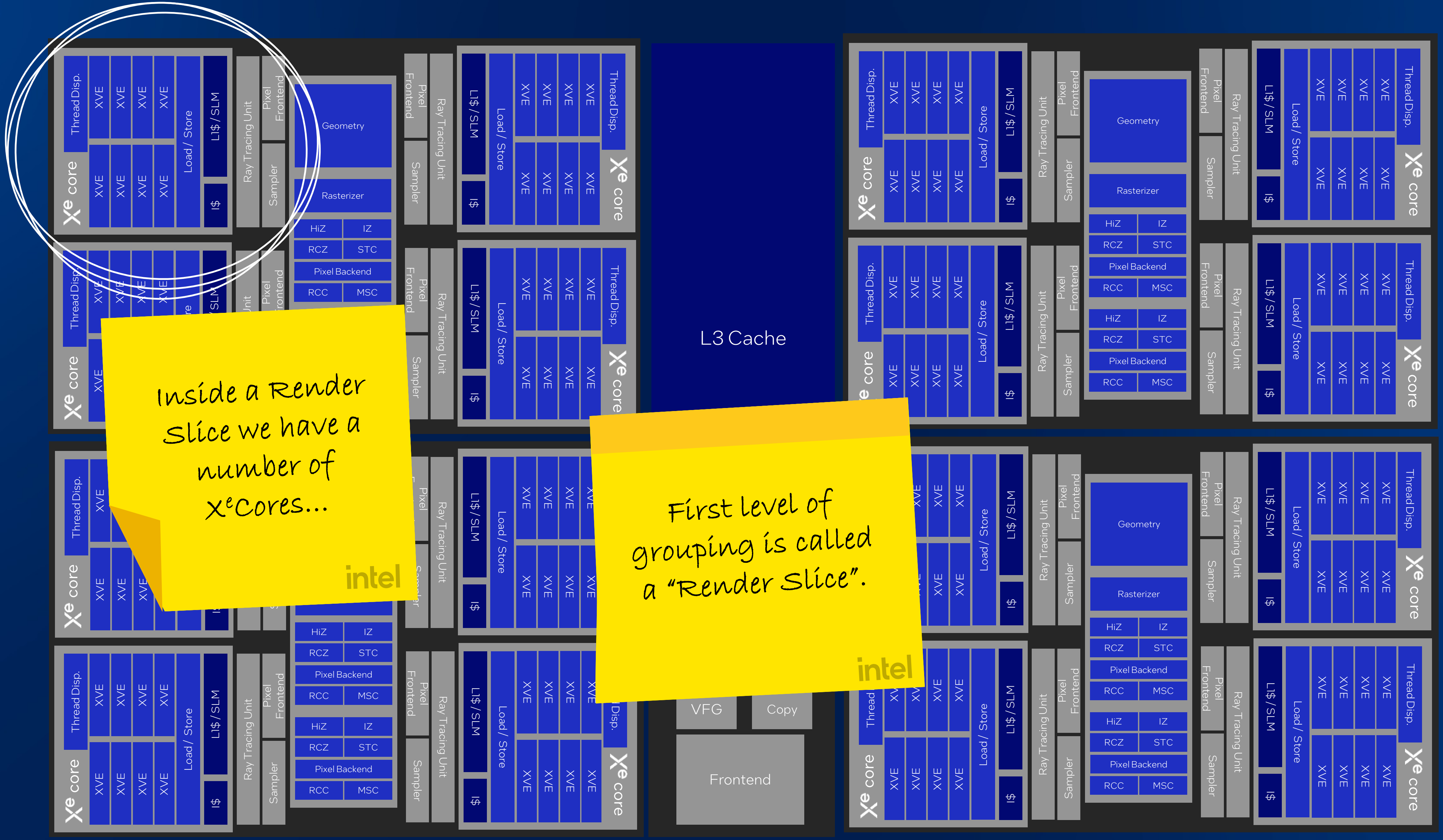
At 5.246 ms
Render Command Streamer Utilization (%) 99.00
Thread Dispatcher Utilization (%) 99.00
XVE Utilization (%) 92.00
GPU Memory Utilization (%) 89.00
Sampler Utilization (%) 50.00
Level 3 Cache Utilization (%) 2.00
Load Store Cache Utilization (%) 1.00
Rasterizer Utilization (%) 0.00
Clipper Utilization (%) 0.00
Color Utilization (%) 0.00
Vertex Fetch Utilization (%) 0.00
Z Utilization (%) 0.00
Compute Copy Engine Utilization (%) 0.00
Compute Command Streamer Utilization (%) 0.00

Might as well
start here, but
first...


Thread Disp.
XVE
XVE
XVE
XVE
XVE
XVE
XVE
XVE
Load / Store
L1$ / SLM
I$
Xe core
Ray Tracing Unit
Pixel Frontend
Sampler
Geometry
Rasterizer
HiZ
IZ
RCZ
STC
Pixel Backend
RCC
MSC
HiZ
IZ
RCZ
STC
Pixel Backend
RCC
MSC
Pixel Frontend
Sampler
Ray Tracing Unit
L1$ / SLM
I$
Load / Store
XVE
XVE
XVE
XVE
XVE
XVE
XVE
XVE
Thread Disp.
Xe core
L3 Cache
Global
VFG
Copy
Frontend

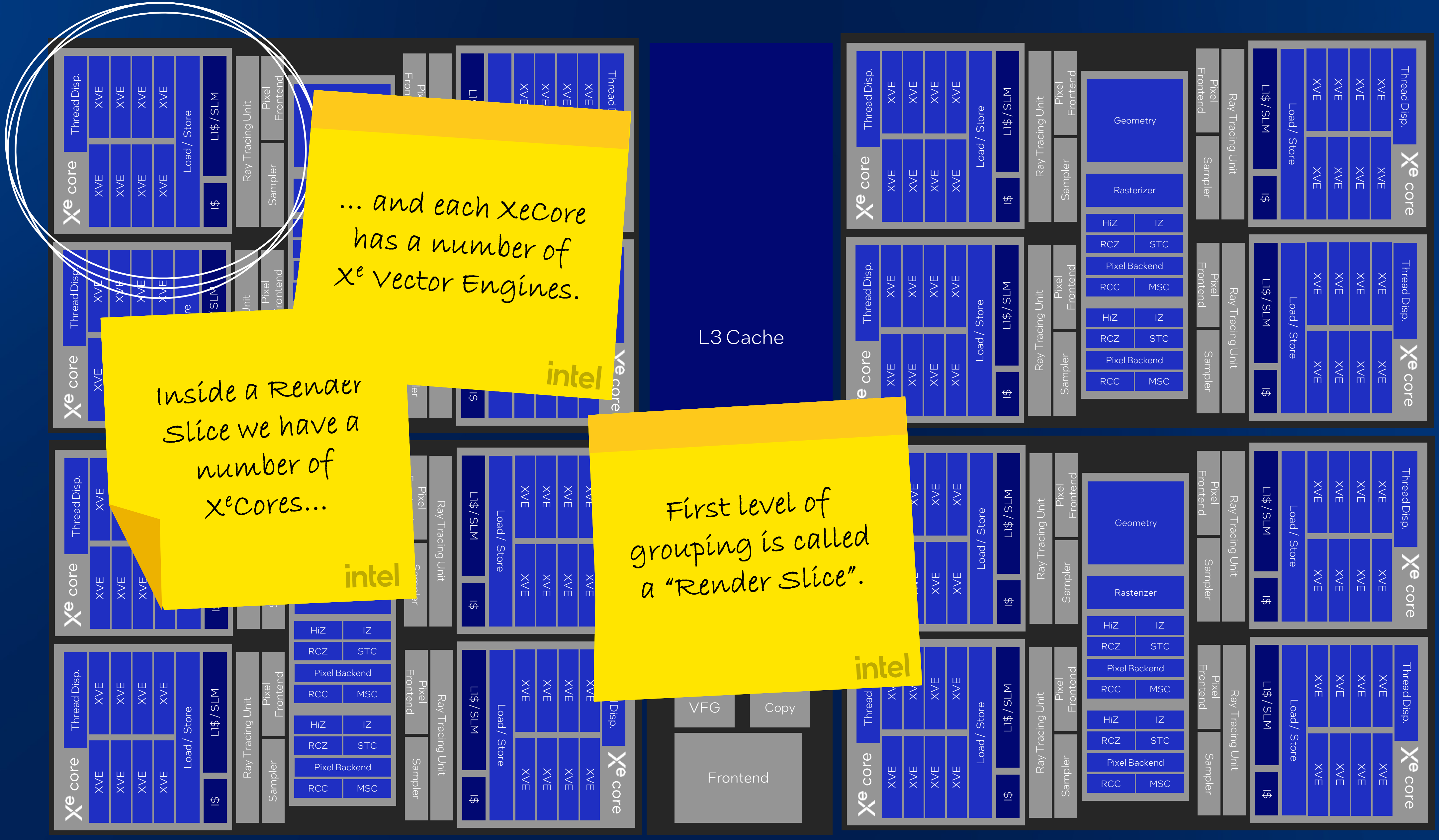


First level of grouping is called a "Render Slice".



Inside a Render Slice we have a number of X^eCores...

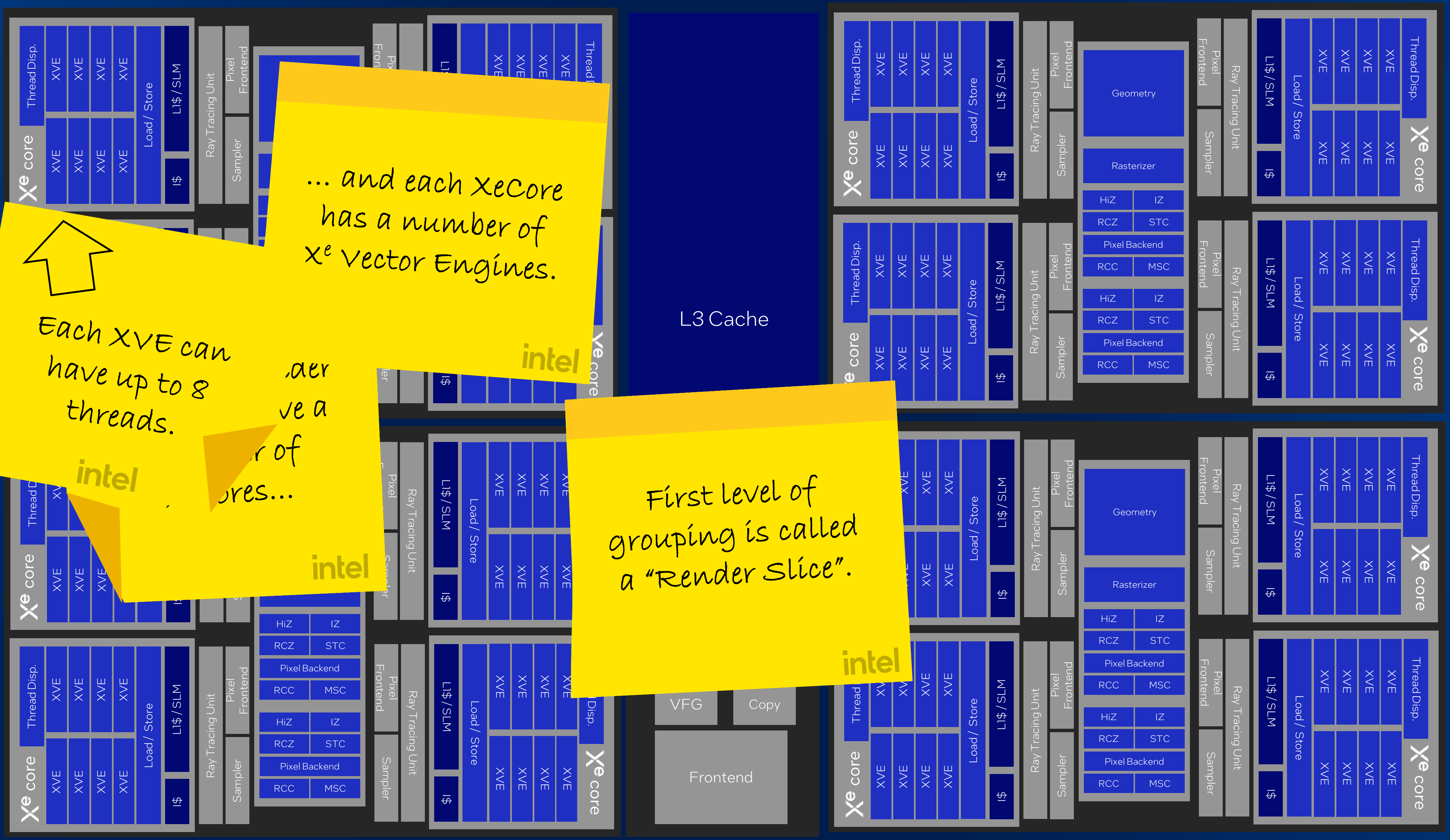
First level of grouping is called a "Render Slice".



First level of grouping is called a "Render Slice".

Inside a Render Slice we have a number of X^eCores...

... and each XeCore has a number of X^e Vector Engines.

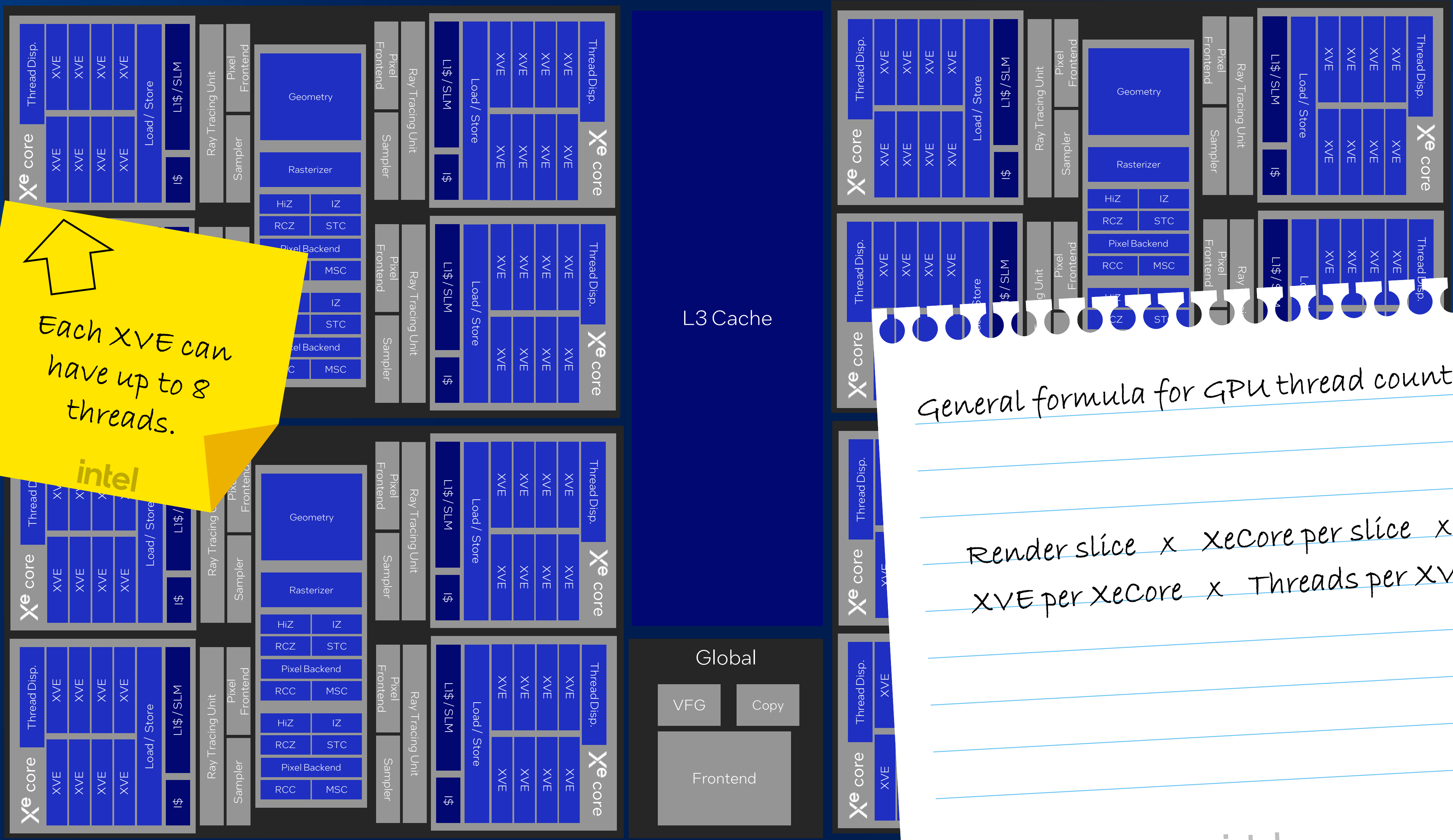


… and each XeCore has a number of X^e Vector Engines.

Each XVE can have up to 8 threads.

,aer
ve a
r of
ores…

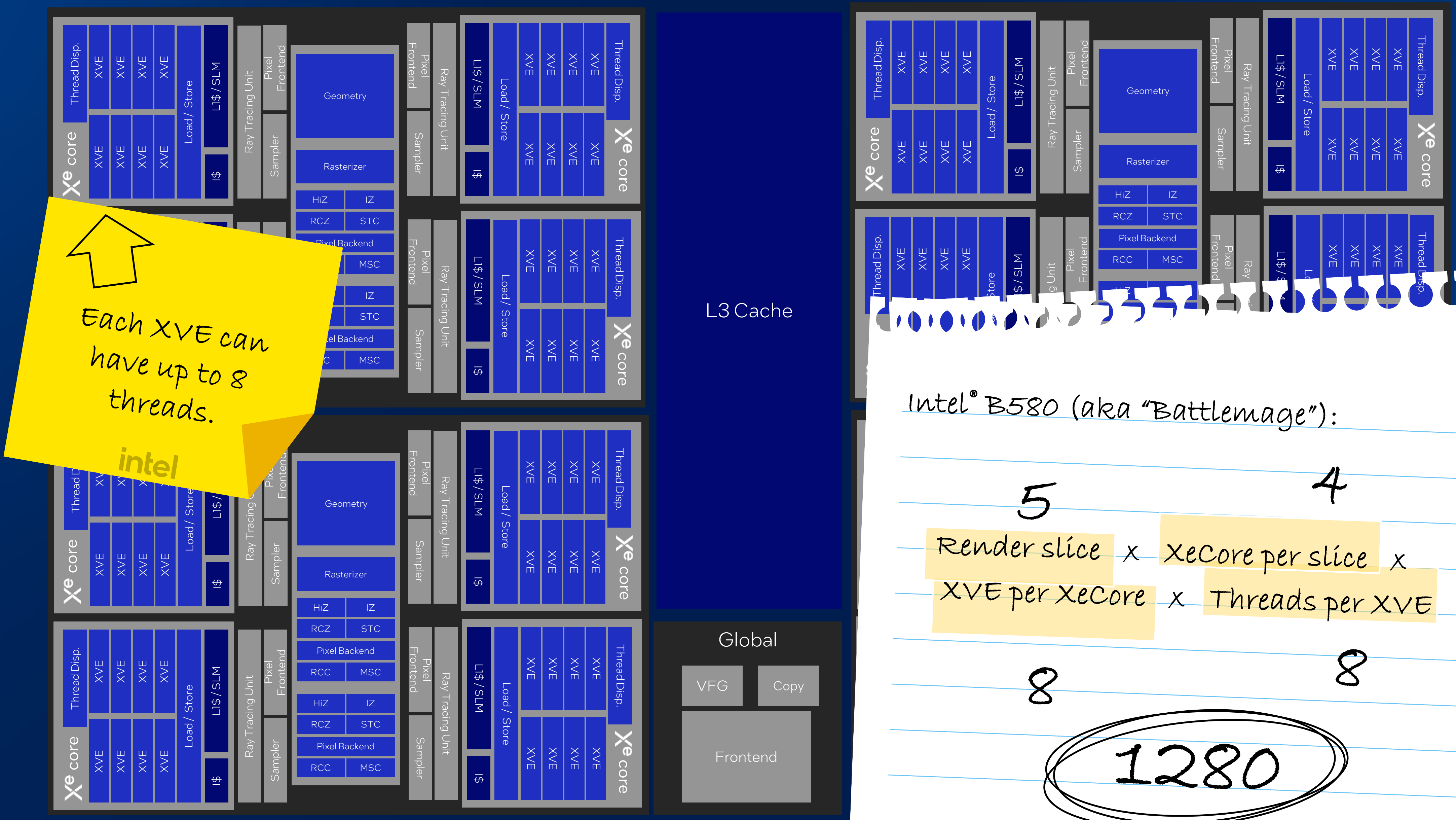
First level of grouping is called a "Render Slice".



Each XVE can have up to 8 threads.

General formula for GPU thread count:

Render slice x XeCore per slice x
XVE per XeCore x Threads per XVE



Each XVE can have up to 8 threads.

Intel® B580 (aka "Battlemage"):

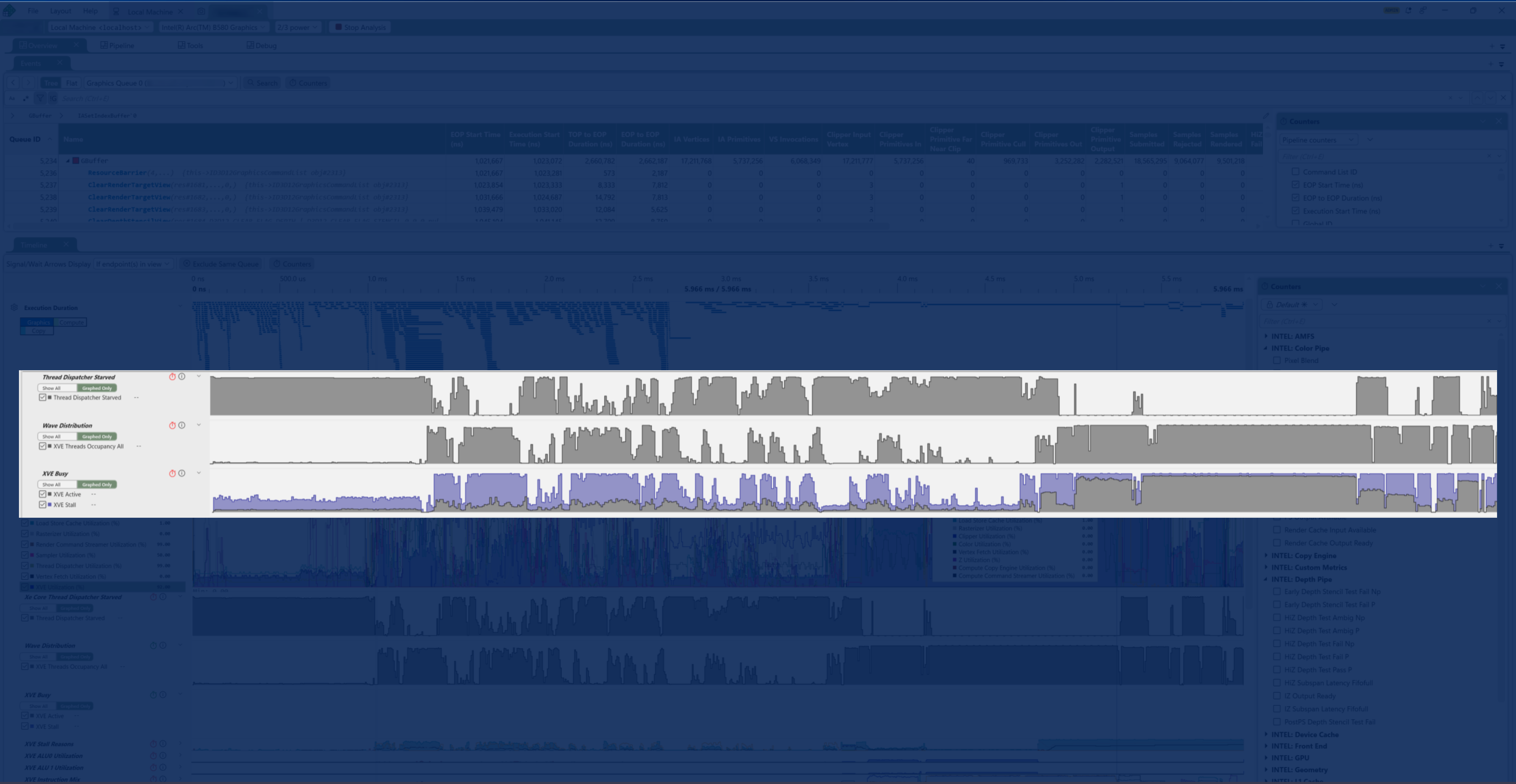
5 4

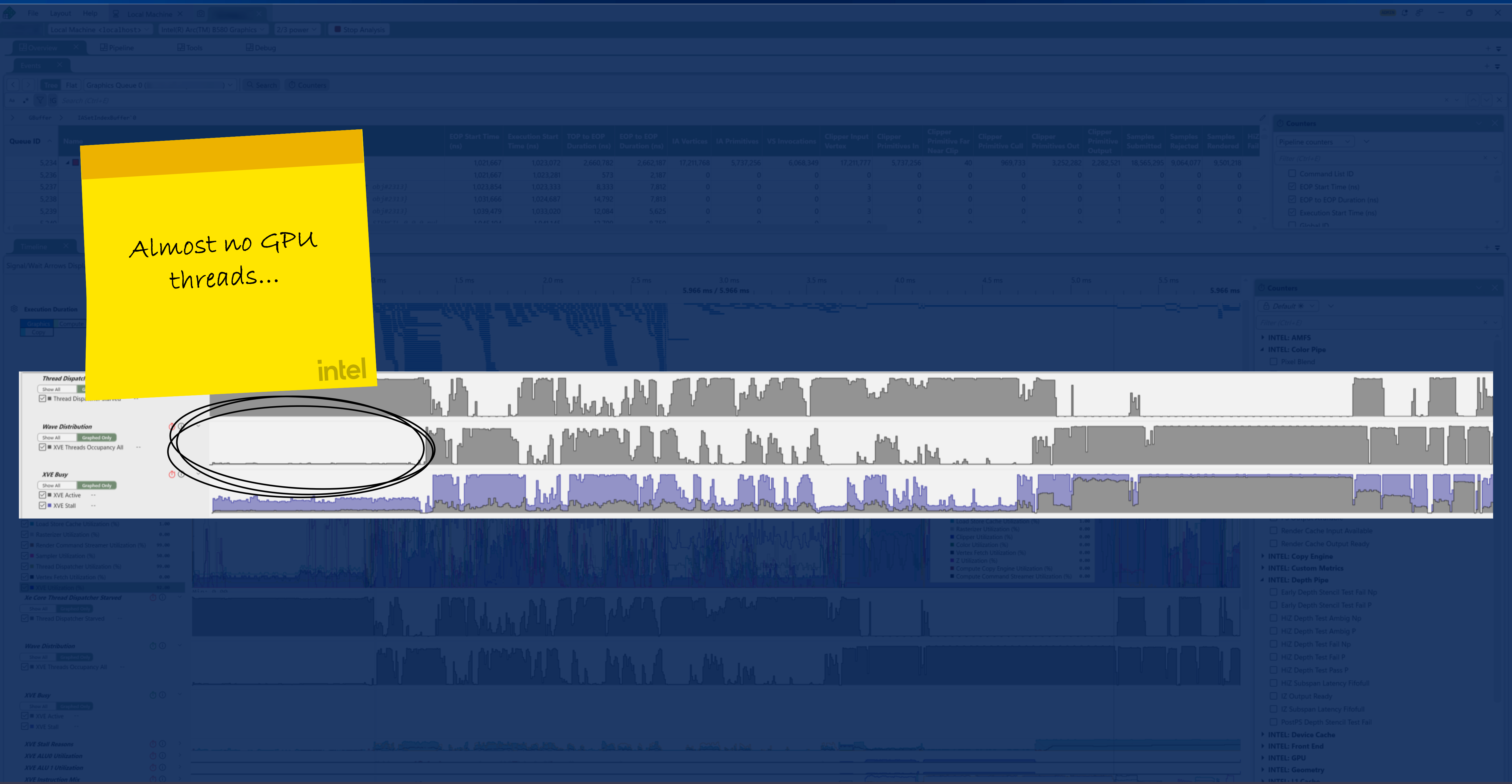
Render slice x XeCore per slice x

XVE per XeCore x Threads per XVE

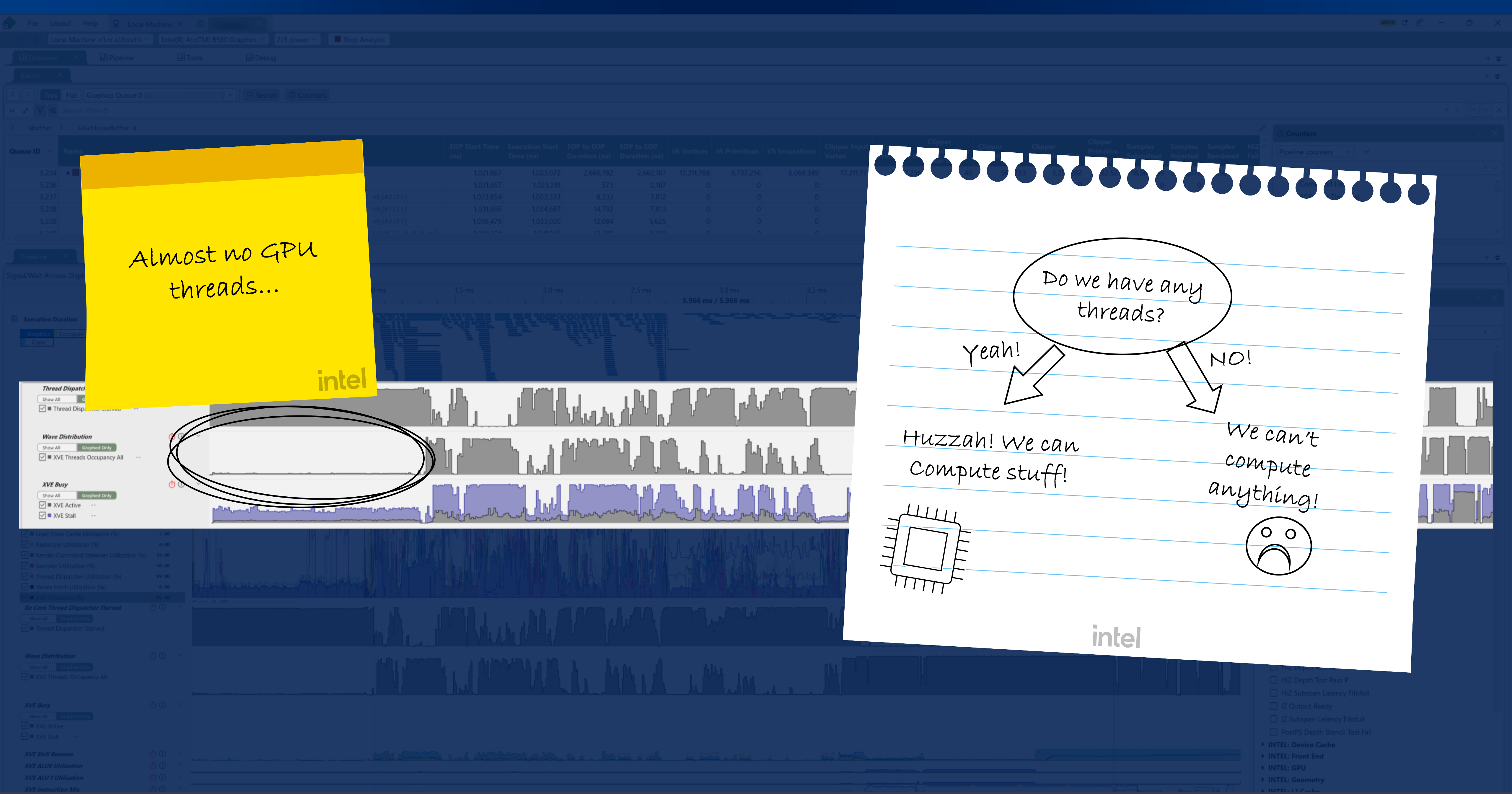
8 8

1280


Thread Dispatcher Starved
Show All
Graphed Only
☑ Thread Dispatcher Starved --
Wave Distribution
Show All
Graphed Only
☑ XVE Threads Occupancy All --
XVE Busy
Show All
Graphed Only
☑ XVE Active --
☑ XVE Stall --



Almost no GPU
threads...



Almost no GPU threads...

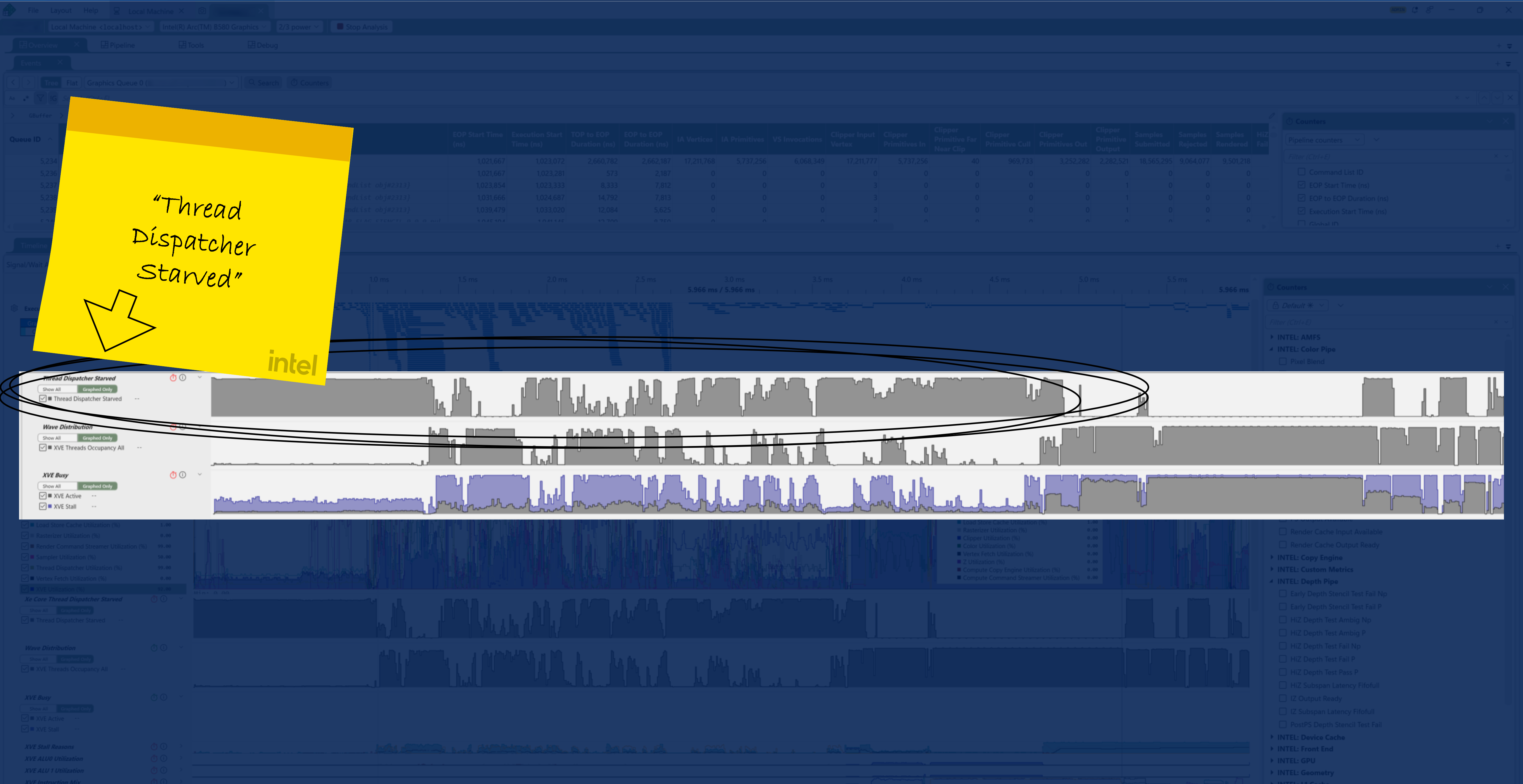
Do we have any threads?

Yeah!

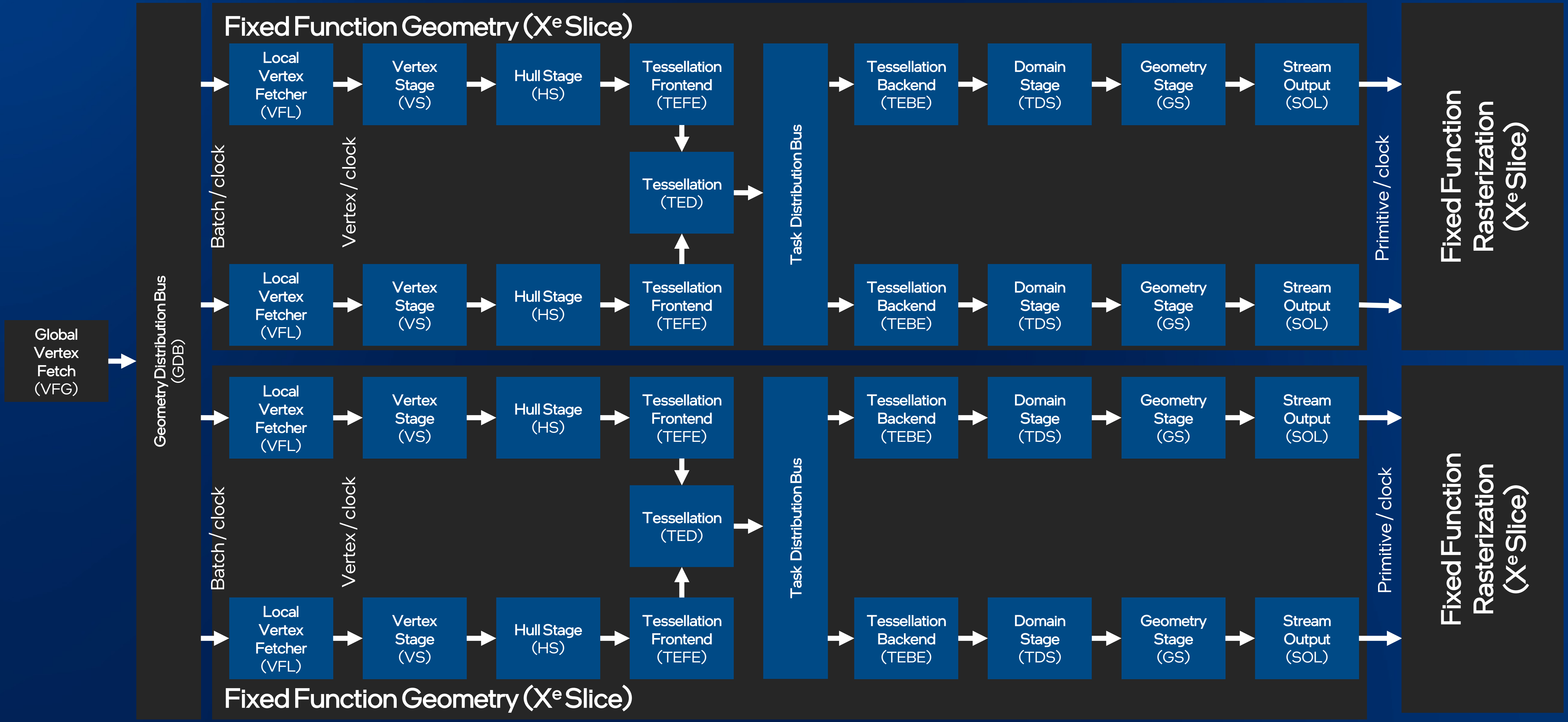
NO!

Huzzah! We can Compute stuff!

We can't compute anything!



"Thread Dispatcher Starved"



Global Vertex Fetch (VFG)

Geometry Distribution Bus (GDB)

Fixed Function Geometry (X^e Slice)

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Batch / clock

Vertex / clock

Tessellation (TED)

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Task Distribution Bus

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Primitive / clock

Fixed Function Rasterization (X^e Slice)

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Batch / clock

Vertex / clock

Tessellation (TED)

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Task Distribution Bus

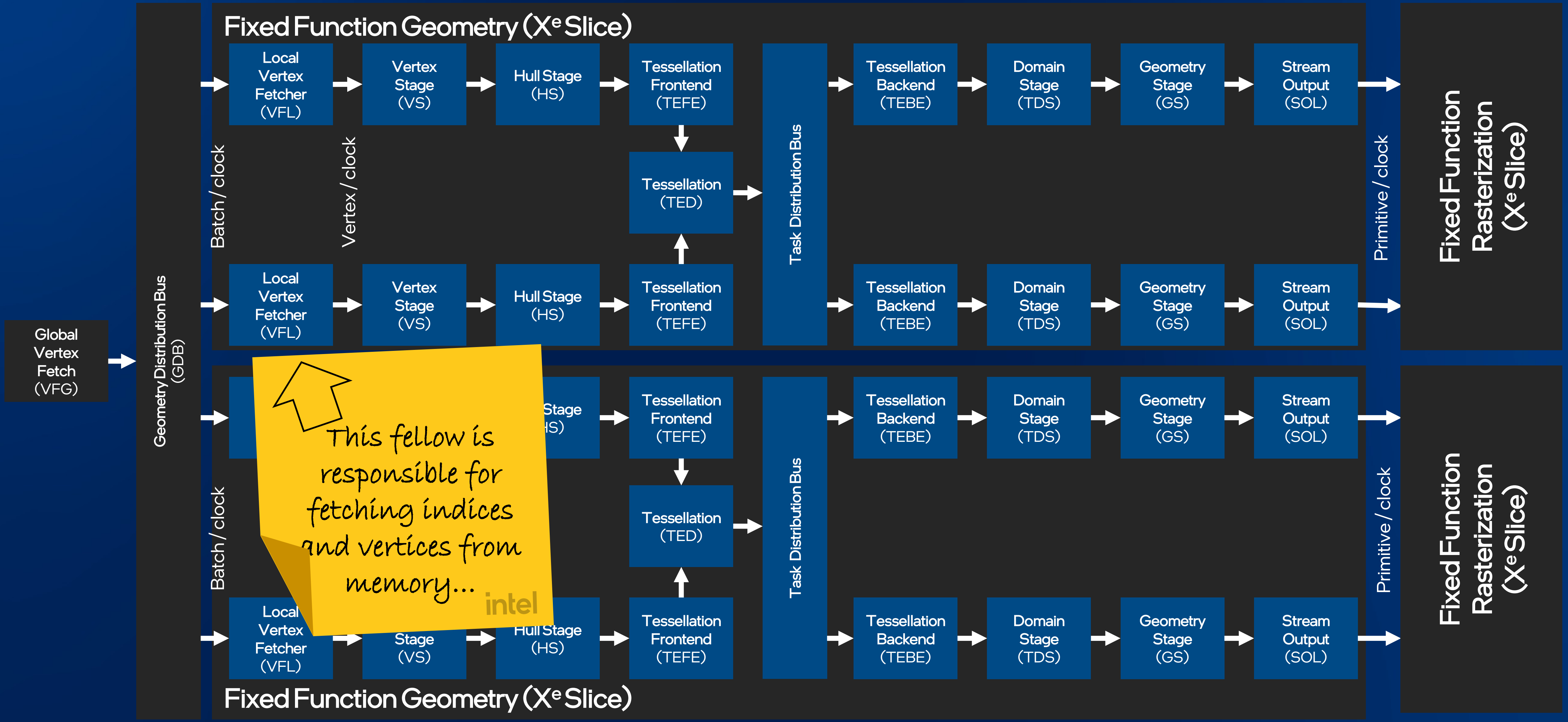
Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Primitive / clock

Fixed Function Rasterization (X^e Slice)

Fixed Function Geometry (X^e Slice)



Global Vertex Fetch (VFG)

Geometry Distribution Bus (GDB)

Fixed Function Geometry (X^e Slice)

Batch / clock

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Vertex / clock

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Tessellation (TED)

Task Distribution Bus

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Primitive / clock

Fixed Function Rasterization (X^e Slice)

Batch / clock

Stage (HS) → Tessellation Frontend (TEFE)

Local Vertex Fetcher (VFL) → Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE)

Tessellation (TED)

Task Distribution Bus

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

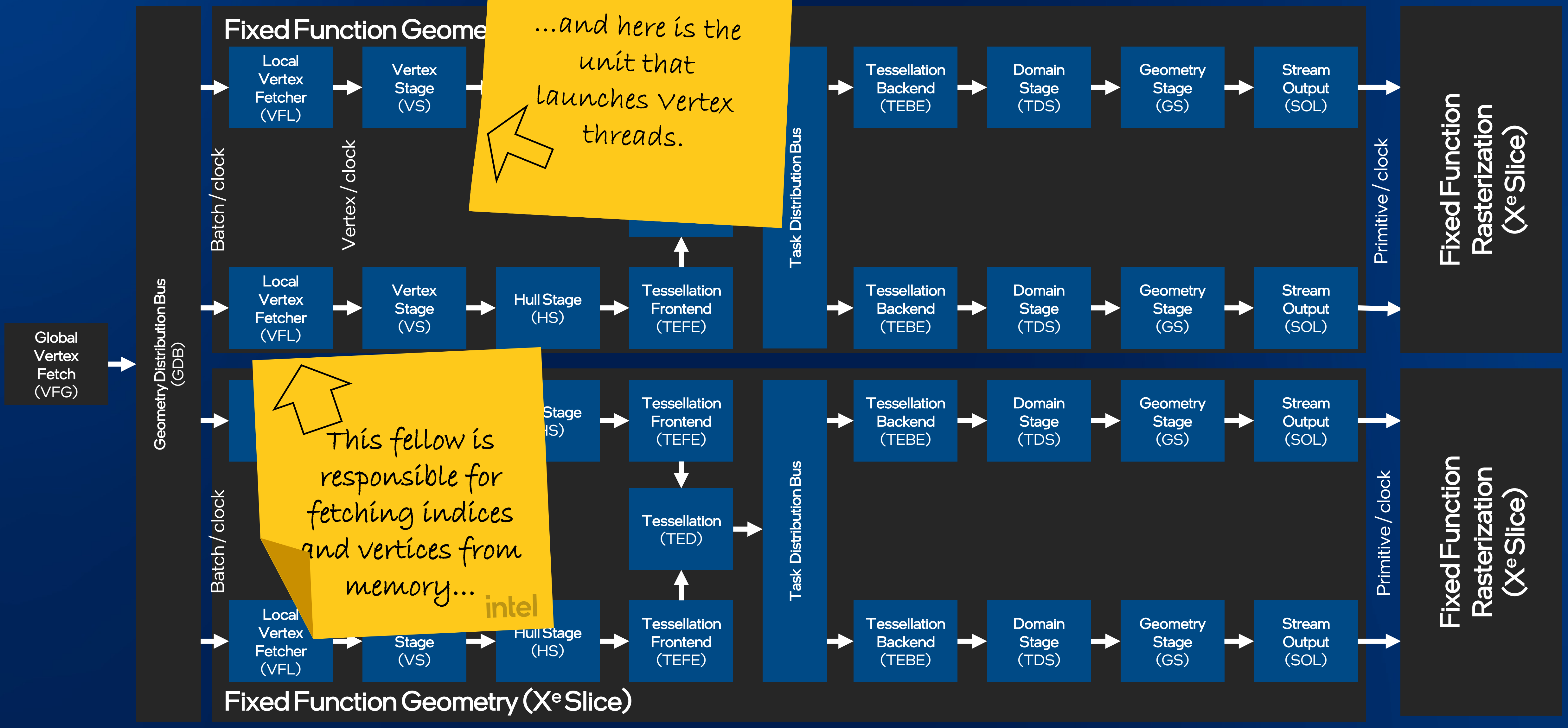
Primitive / clock

Fixed Function Rasterization (X^e Slice)

Fixed Function Geometry (X^e Slice)

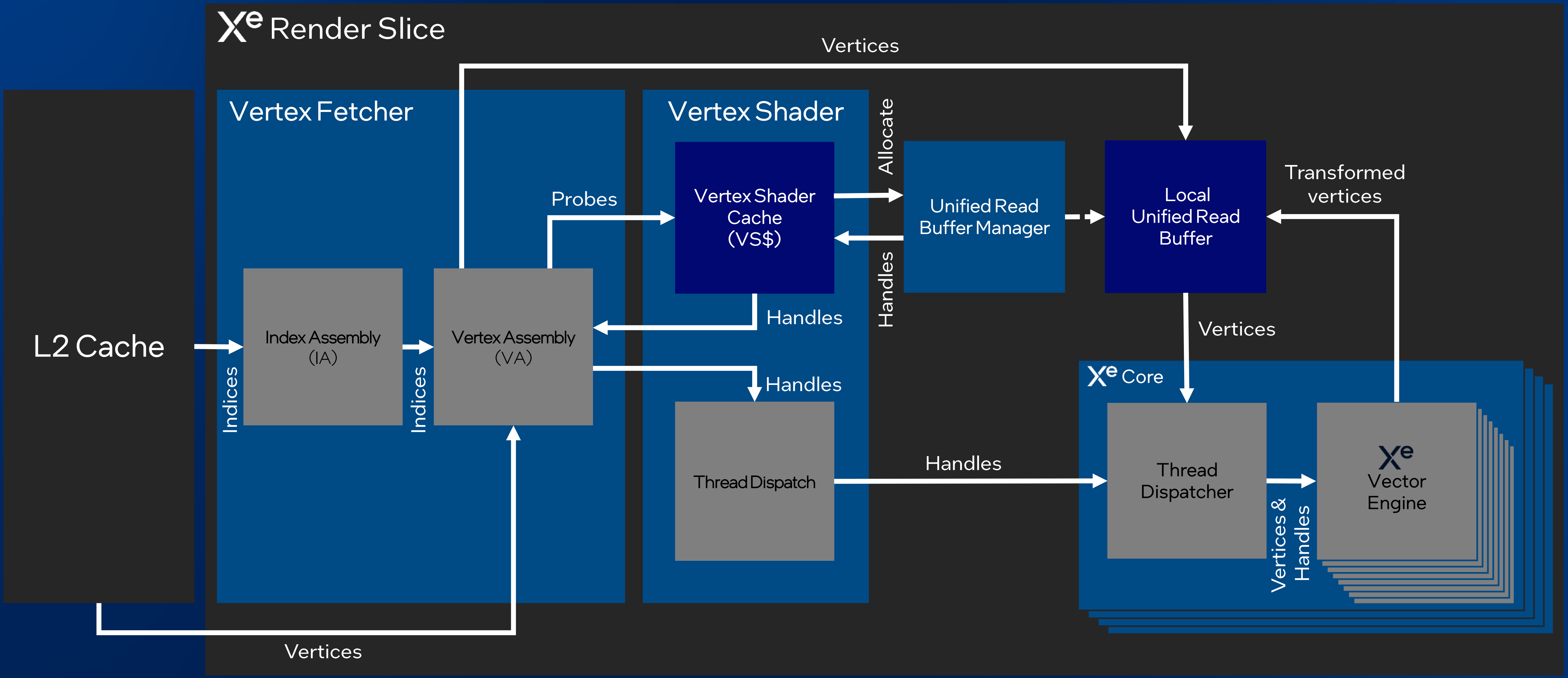
This fellow is responsible for fetching indices and vertices from memory...

intel



...and here is the unit that launches vertex threads.

This fellow is responsible for fetching indices and vertices from memory...



Xe Render Slice

L2 Cache

Vertex Fetcher

Index Assembly (IA)

Vertex Assembly (VA)

Vertex Shader

Vertex Shader Cache (VS$)

Thread Dispatch

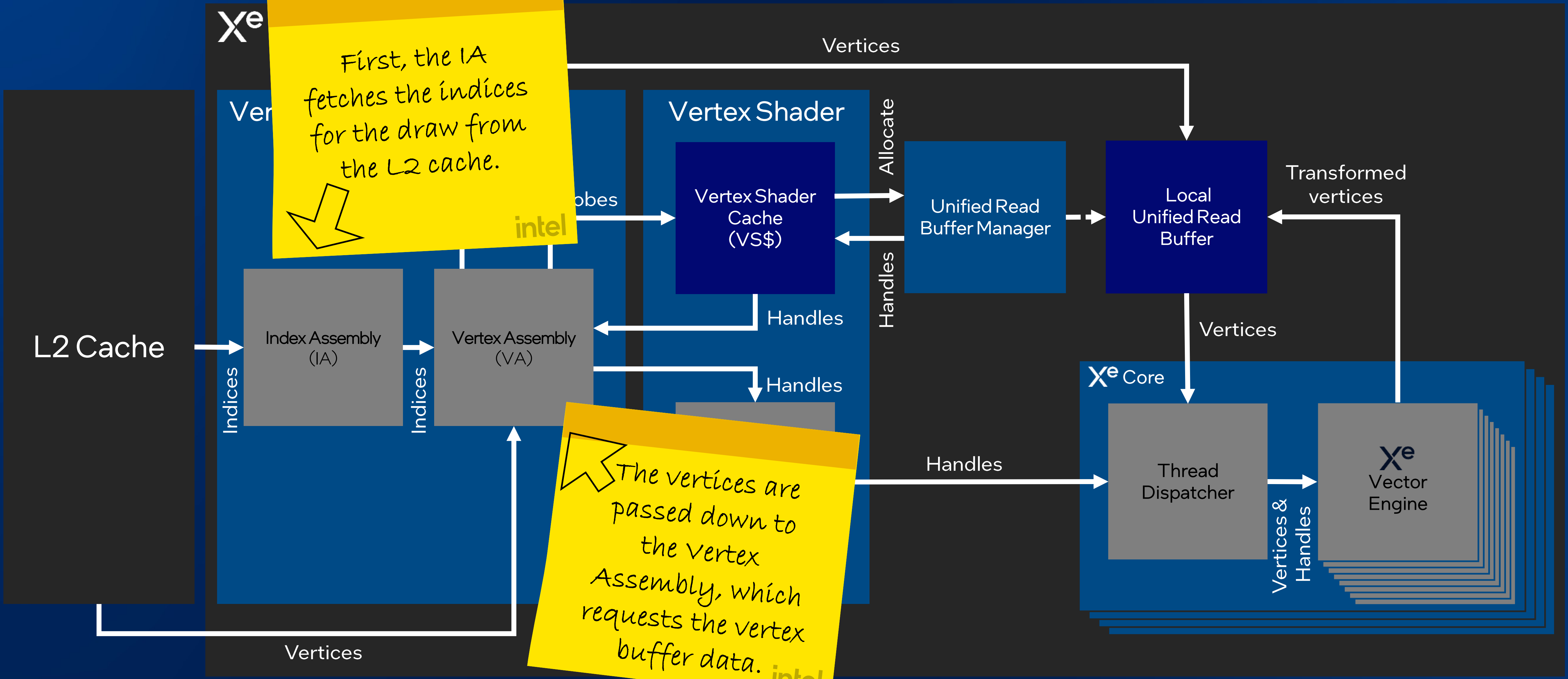
Unified Read Buffer Manager

Local Unified Read Buffer

Xe Core

Thread Dispatcher

Xe Vector Engine

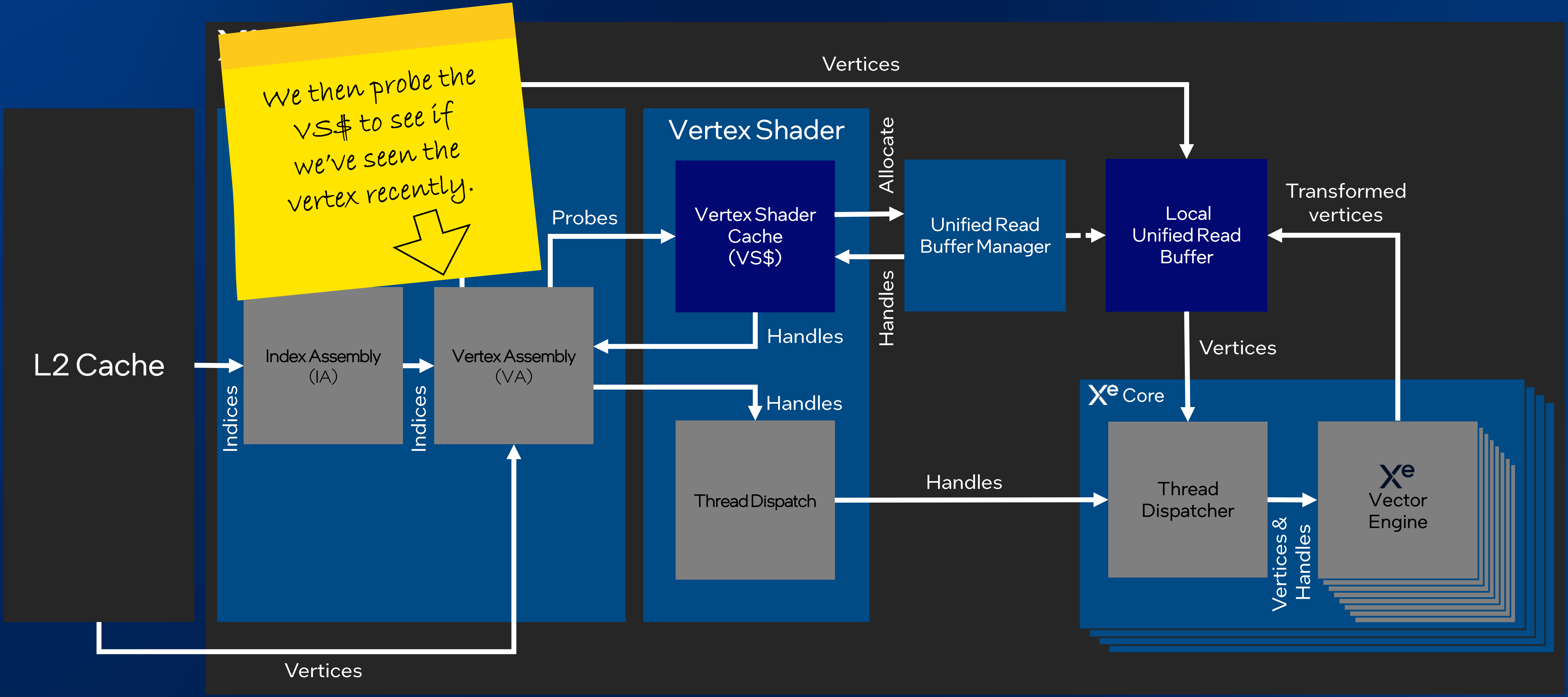


First, the IA fetches the indices for the draw from the L2 cache.

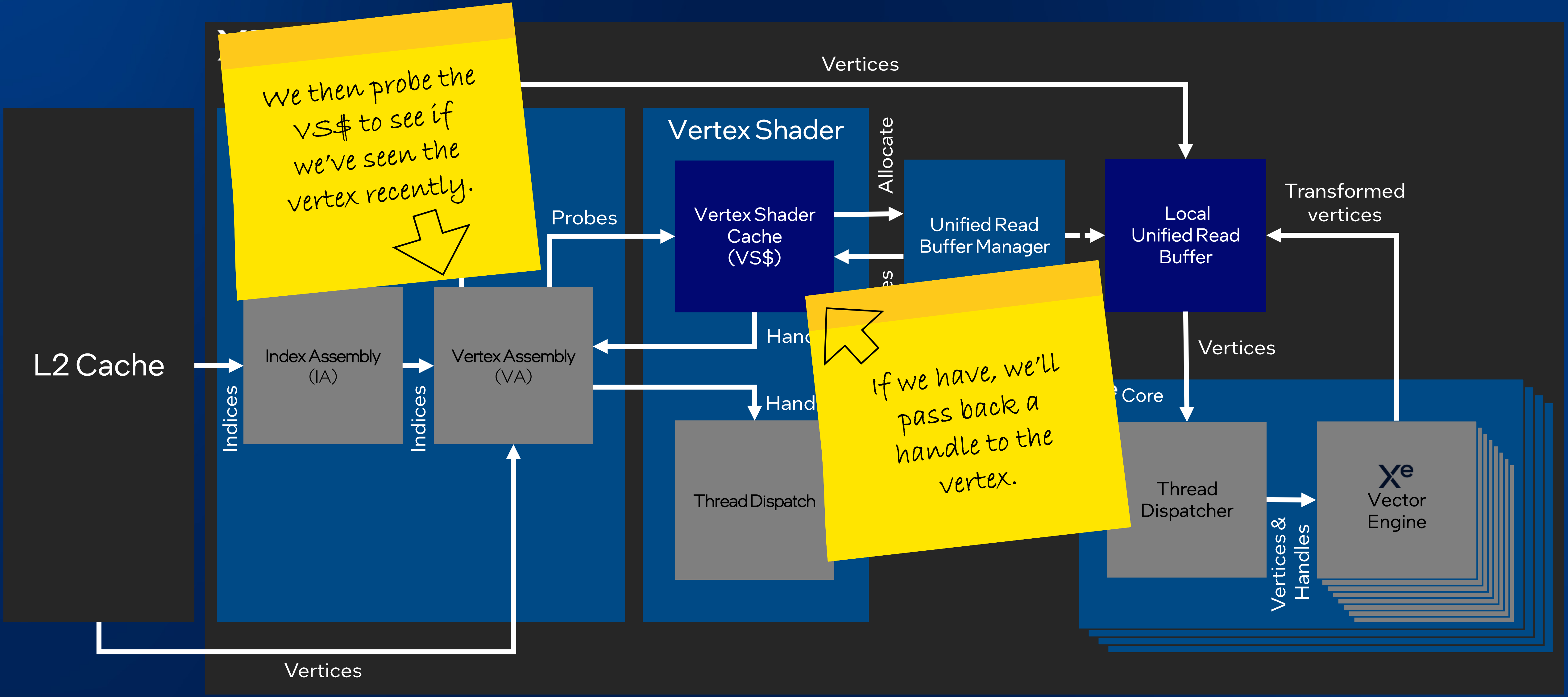
intel

The vertices are passed down to the Vertex Assembly, which requests the vertex buffer data.

intel

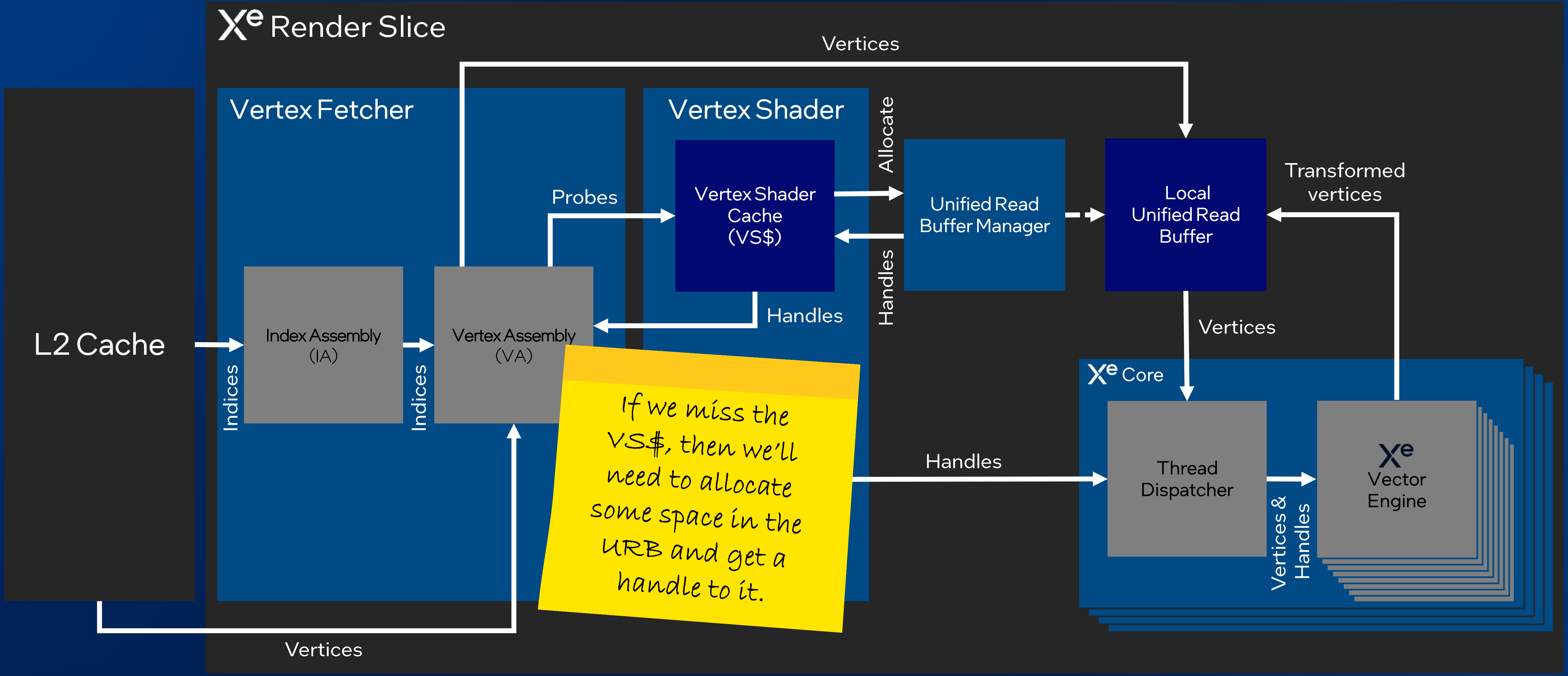


We then probe the
VS$ to see if
we've seen the
vertex recently.

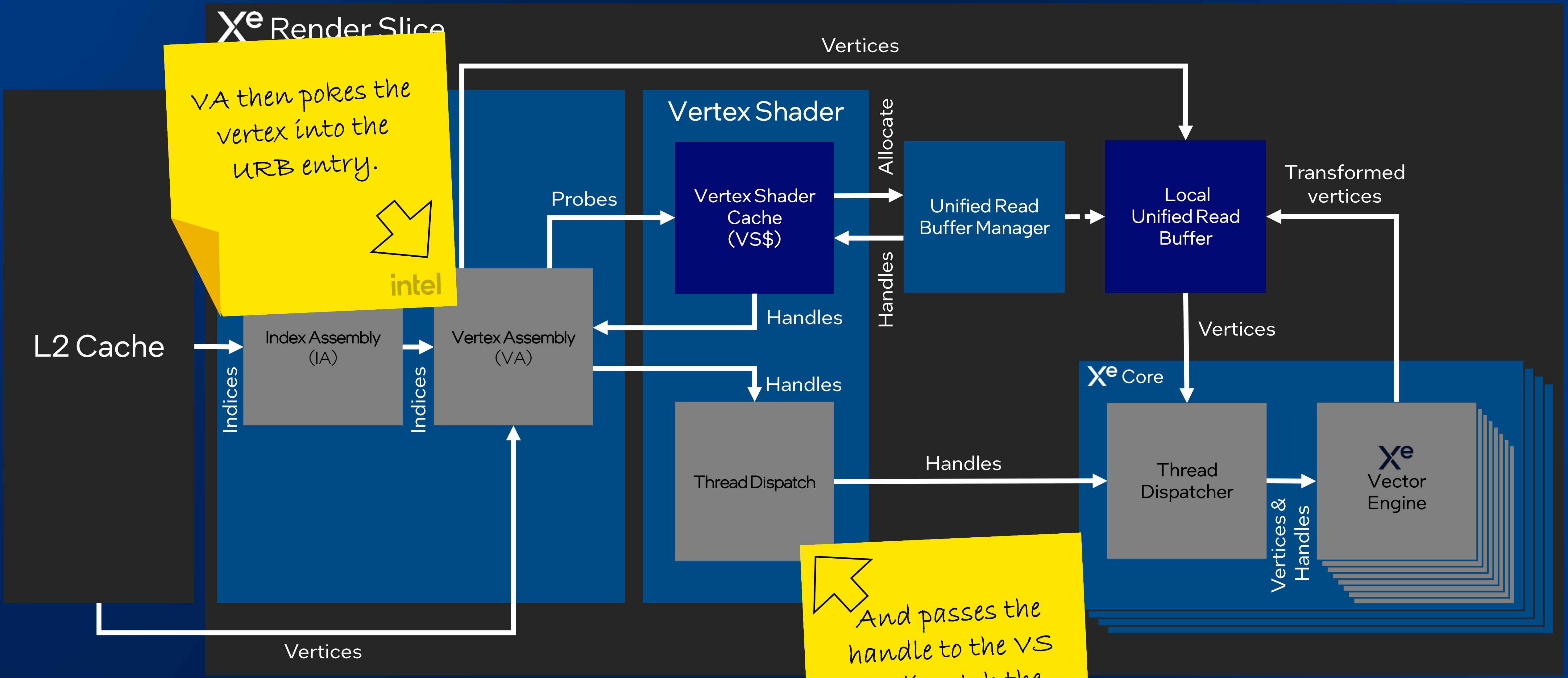


We then probe the VS$ to see if we've seen the vertex recently.

If we have, we'll pass back a handle to the vertex.

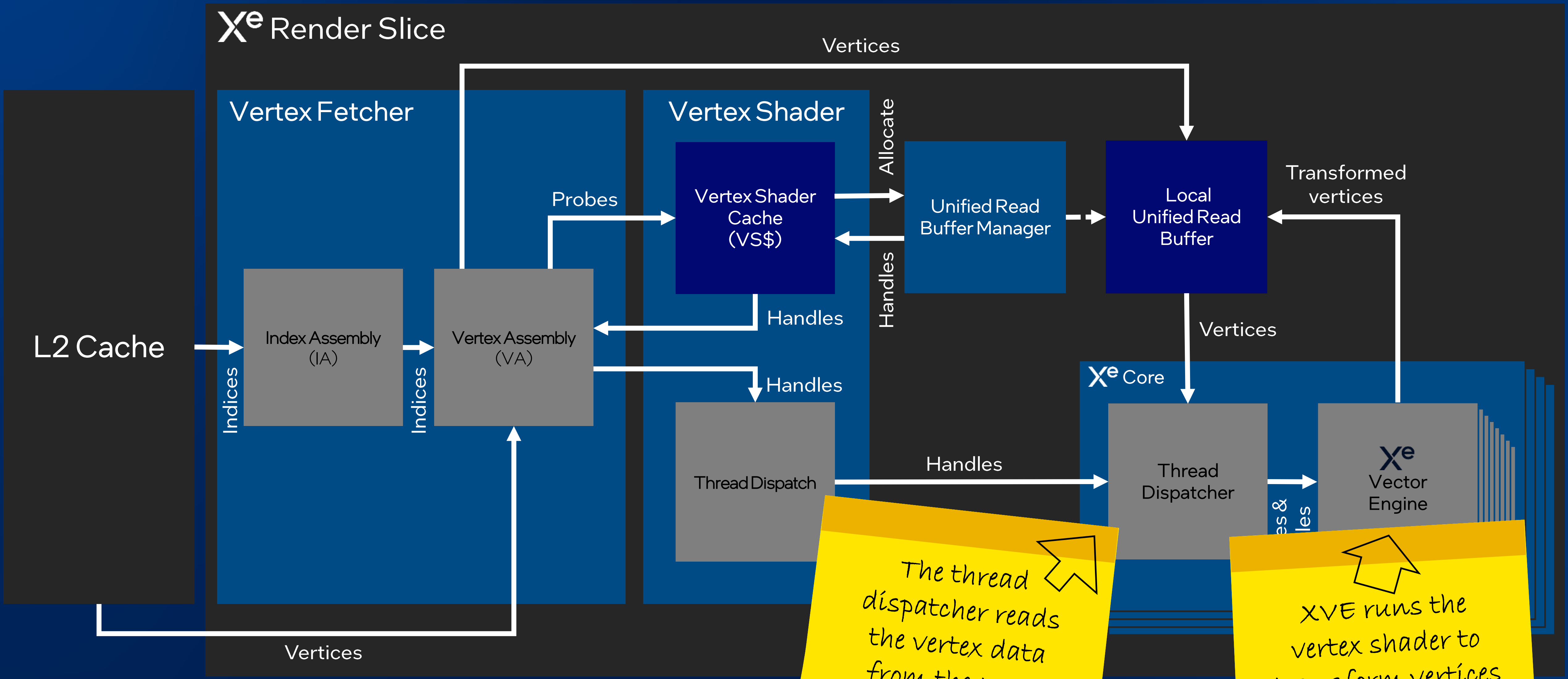


If we miss the VS$, then we'll need to allocate some space in the URB and get a handle to it.



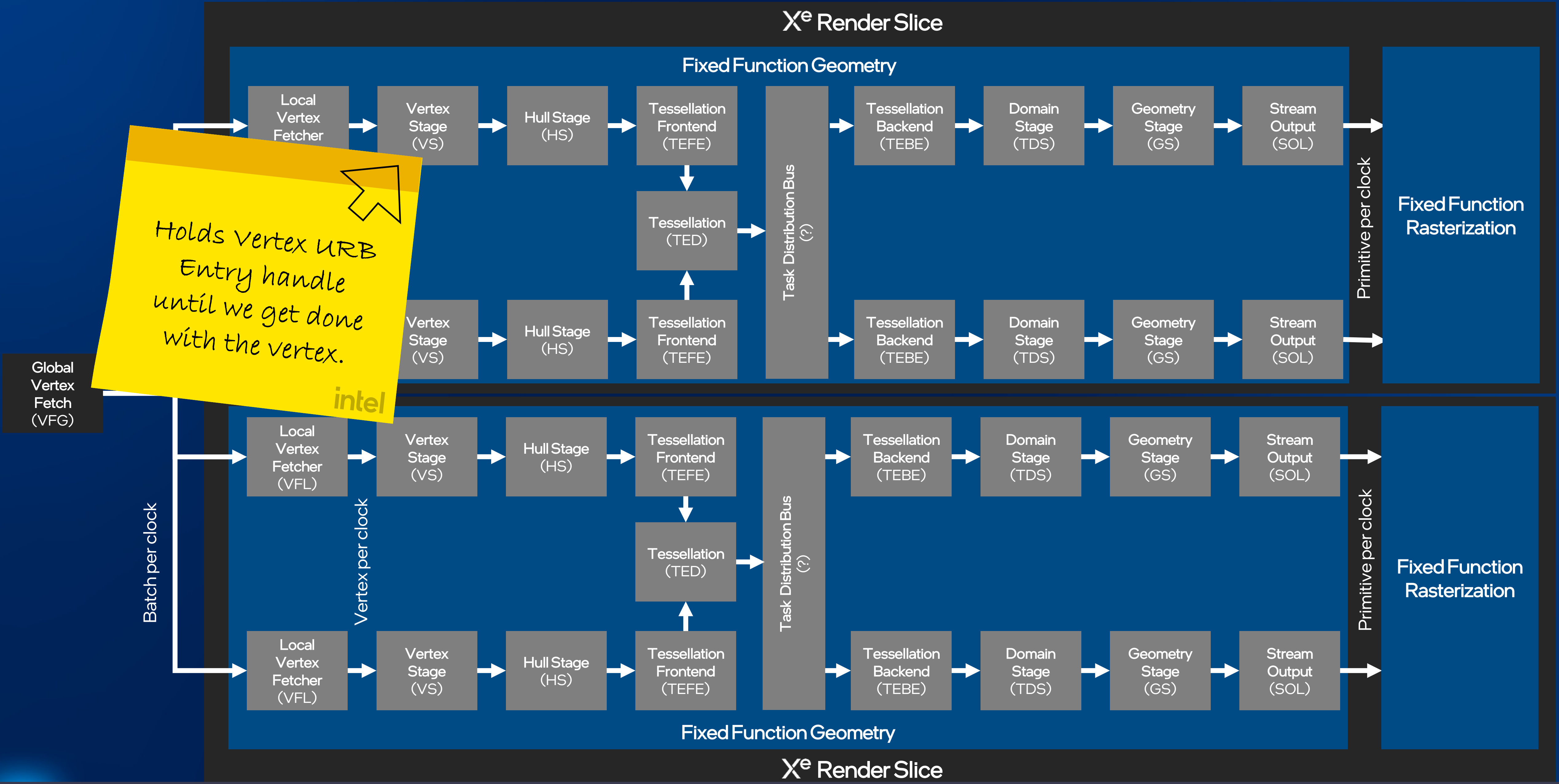
VA then pokes the vertex into the URB entry.

And passes the handle to the VS to dispatch the thread.



The thread dispatcher reads the vertex data from the URB and launches the thread.

XVE runs the vertex shader to transform vertices and writes back to the URB.



X^e Render Slice

Fixed Function Geometry

Local Vertex Fetcher → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE) → Task Distribution Bus (?) → Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation (TED)

Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE) → Task Distribution Bus (?) → Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Primitive per clock

Fixed Function Rasterization

Holds Vertex URB Entry handle until we get done with the vertex.

intel

Global Vertex Fetch (VFG)

Batch per clock

Vertex per clock

Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE) → Task Distribution Bus (?) → Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Tessellation (TED)

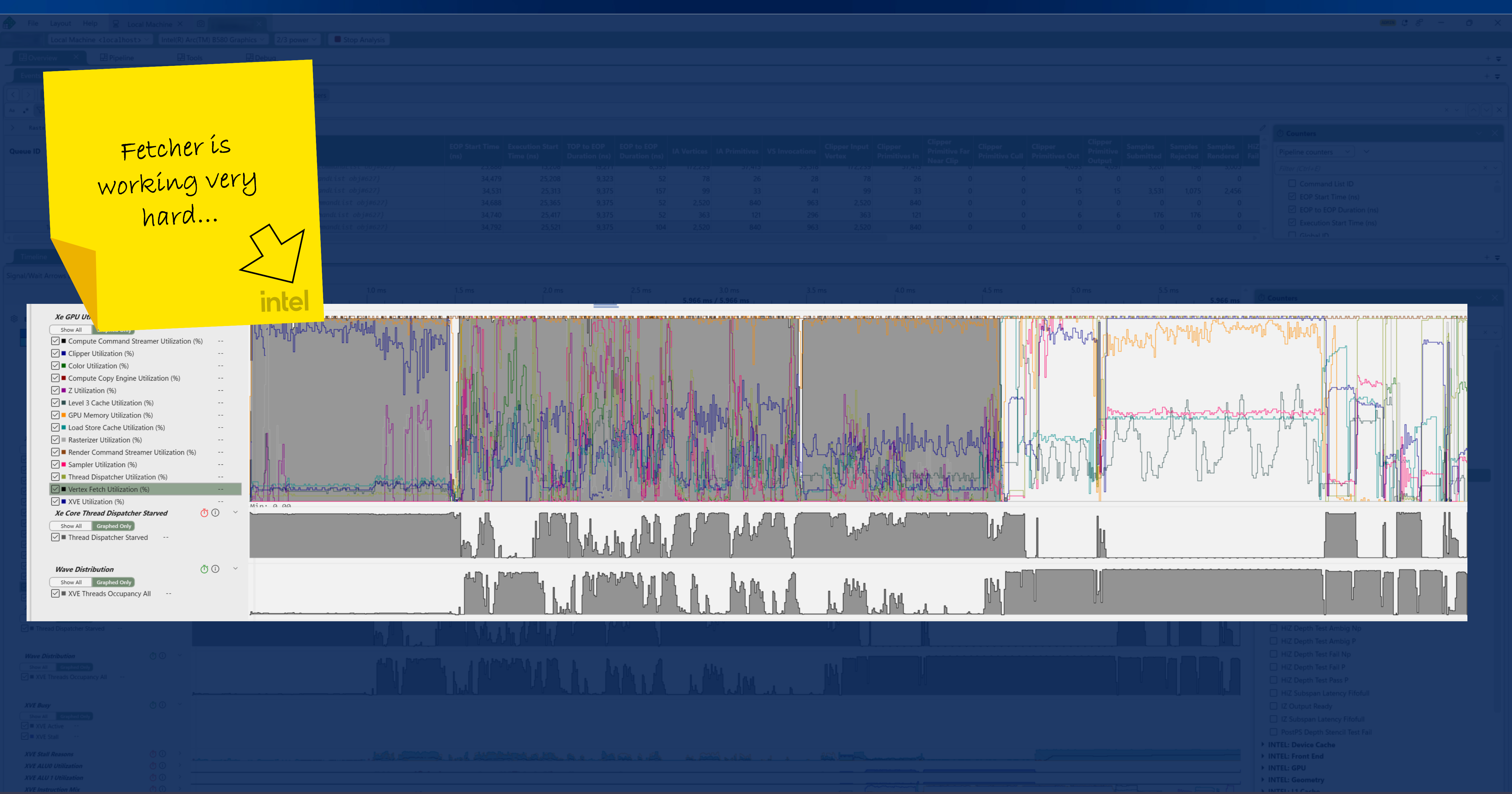
Local Vertex Fetcher (VFL) → Vertex Stage (VS) → Hull Stage (HS) → Tessellation Frontend (TEFE) → Task Distribution Bus (?) → Tessellation Backend (TEBE) → Domain Stage (TDS) → Geometry Stage (GS) → Stream Output (SOL)

Primitive per clock

Fixed Function Rasterization

Fixed Function Geometry

X^e Render Slice



Fetcher is working very hard...

Xe GPU Ut

☑ Compute Command Streamer Utilization (%) --
☑ Clipper Utilization (%) --
☑ Color Utilization (%) --
☑ Compute Copy Engine Utilization (%) --
☑ Z Utilization (%) --
☑ Level 3 Cache Utilization (%) --
☑ GPU Memory Utilization (%) --
☑ Load Store Cache Utilization (%) --
☑ Rasterizer Utilization (%) --
☑ Render Command Streamer Utilization (%) --
☑ Sampler Utilization (%) --
☑ Thread Dispatcher Utilization (%) --
☑ Vertex Fetch Utilization (%) --
☑ XVE Utilization (%) --

Xe Core Thread Dispatcher Starved

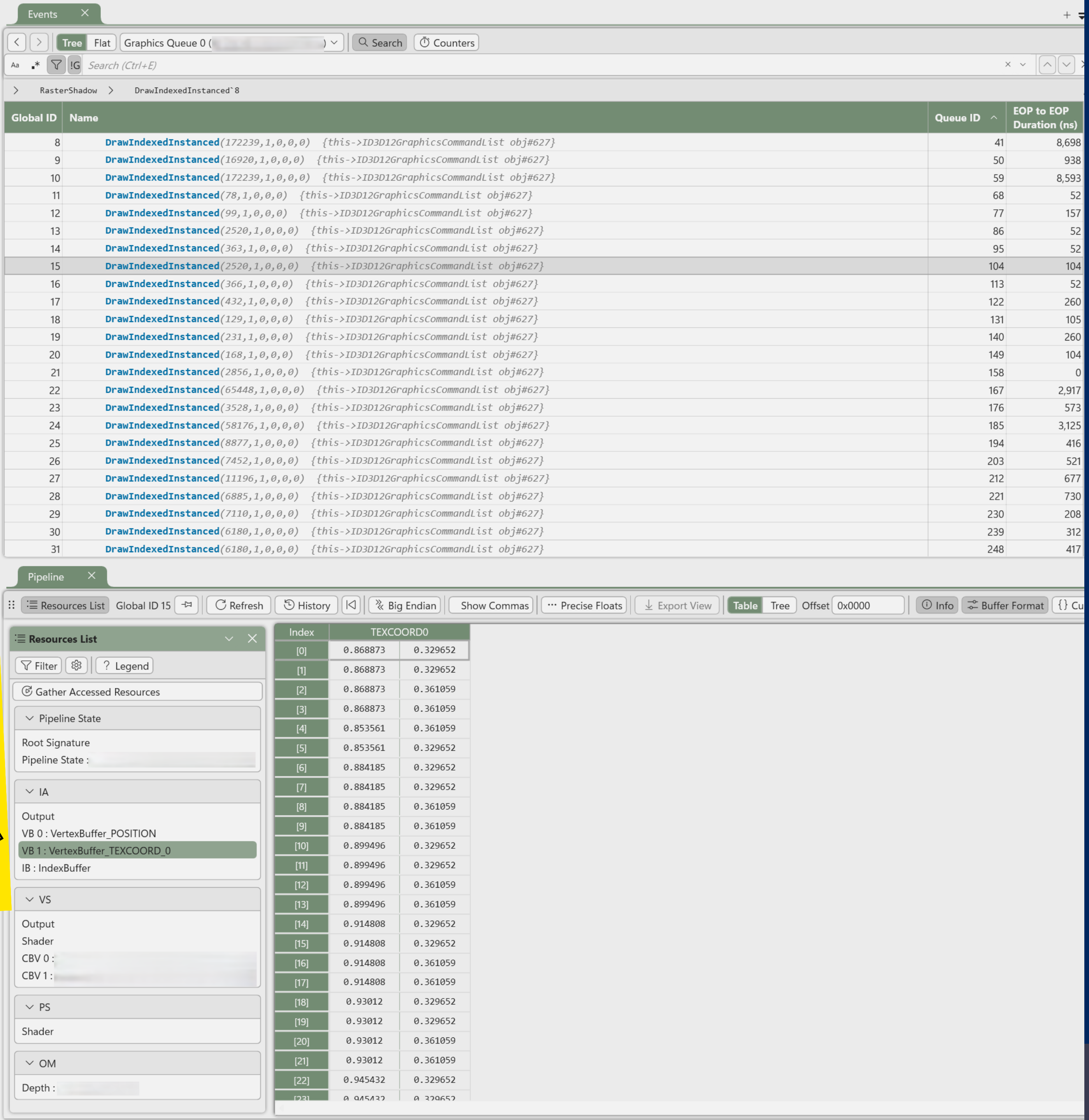
☑ Thread Dispatcher Starved --

Wave Distribution

☑ XVE Threads Occupancy All --

...fetching extraneous vertex data...

intel



RasterShadow > DrawIndexedInstanced`8

Global ID	Name	Queue ID	EOP to EOP Duration (ns)
8	DrawIndexedInstanced(172239,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	41	8,698
9	DrawIndexedInstanced(16920,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	50	938
10	DrawIndexedInstanced(172239,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	59	8,593
11	DrawIndexedInstanced(78,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	68	52
12	DrawIndexedInstanced(99,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	77	157
13	DrawIndexedInstanced(2520,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	86	52
14	DrawIndexedInstanced(363,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	95	52
15	DrawIndexedInstanced(2520,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	104	104
16	DrawIndexedInstanced(366,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	113	52
17	DrawIndexedInstanced(432,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	122	260
18	DrawIndexedInstanced(129,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	131	105
19	DrawIndexedInstanced(231,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	140	260
20	DrawIndexedInstanced(168,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	149	104
21	DrawIndexedInstanced(2856,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	158	0
22	DrawIndexedInstanced(65448,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	167	2,917
23	DrawIndexedInstanced(3528,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	176	573
24	DrawIndexedInstanced(58176,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	185	3,125
25	DrawIndexedInstanced(8877,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	194	416
26	DrawIndexedInstanced(7452,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	203	521
27	DrawIndexedInstanced(11196,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	212	677
28	DrawIndexedInstanced(6885,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	221	730
29	DrawIndexedInstanced(7110,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	230	208
30	DrawIndexedInstanced(6180,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	239	312
31	DrawIndexedInstanced(6180,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	248	417

Global ID 15

Offset 0x0000

Resources List

Pipeline State
- Root Signature
- Pipeline State :

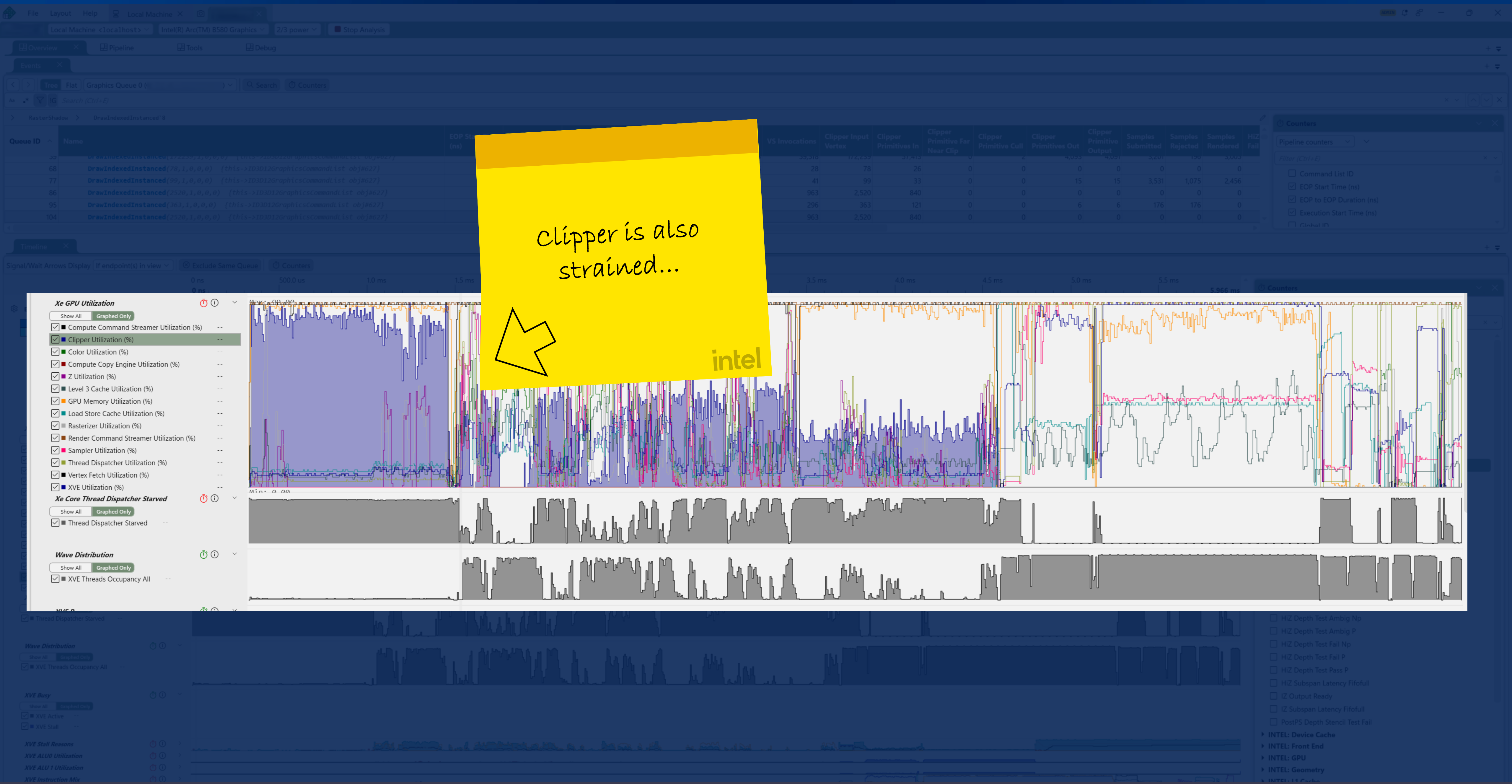
IA
- Output
- VB 0 : VertexBuffer_POSITION
- VB 1 : VertexBuffer_TEXCOORD_0
- IB : IndexBuffer

VS
- Output
- Shader
- CBV 0 :
- CBV 1 :

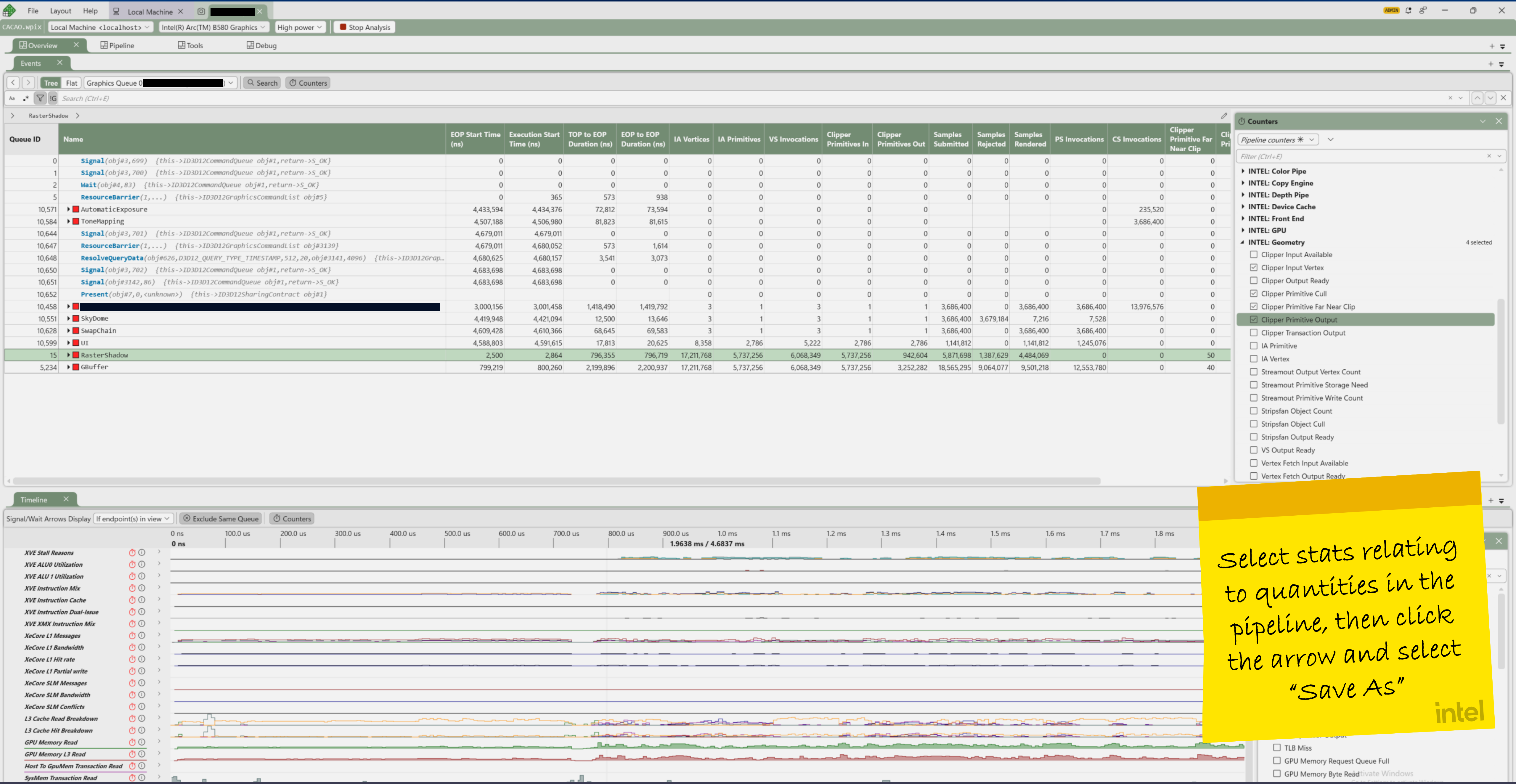
PS
- Shader

OM
- Depth :

Index	TEXCOORD0	
[0]	0.868873	0.329652
[1]	0.868873	0.329652
[2]	0.868873	0.361059
[3]	0.868873	0.361059
[4]	0.853561	0.361059
[5]	0.853561	0.329652
[6]	0.884185	0.329652
[7]	0.884185	0.329652
[8]	0.884185	0.361059
[9]	0.884185	0.361059
[10]	0.899496	0.329652
[11]	0.899496	0.329652
[12]	0.899496	0.361059
[13]	0.899496	0.361059
[14]	0.914808	0.329652
[15]	0.914808	0.329652
[16]	0.914808	0.361059
[17]	0.914808	0.361059
[18]	0.93012	0.329652
[19]	0.93012	0.329652
[20]	0.93012	0.361059
[21]	0.93012	0.361059
[22]	0.945432	0.329652
[23]	0.945432	0.329652


Xe GPU Utilization
Show All
Graphed Only
☑ Compute Command Streamer Utilization (%)
☑ Clipper Utilization (%)
☑ Color Utilization (%)
☑ Compute Copy Engine Utilization (%)
☑ Z Utilization (%)
☑ Level 3 Cache Utilization (%)
☑ GPU Memory Utilization (%)
☑ Load Store Cache Utilization (%)
☑ Rasterizer Utilization (%)
☑ Render Command Streamer Utilization (%)
☑ Sampler Utilization (%)
☑ Thread Dispatcher Utilization (%)
☑ Vertex Fetch Utilization (%)
☑ XVE Utilization (%)
Xe Core Thread Dispatcher Starved
Show All
Graphed Only
☑ Thread Dispatcher Starved
Wave Distribution
Show All
Graphed Only
☑ XVE Threads Occupancy All
Min: 0.00
intel

Clipper is also strained...



Queue ID	Name	EOP Start Time (ns)	Execution Start Time (ns)	TOP to EOP Duration (ns)	EOP to EOP Duration (ns)	IA Vertices	IA Primitives	VS Invocations	Clipper Primitives In	Clipper Primitives Out	Samples Submitted	Samples Rejected	Samples Rendered	PS Invocations	CS Invocations	Clipper Primitive Far Near Clip
0	**Signal**(obj#3,699) {this->ID3D12CommandQueue obj#1,return->S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	**Signal**(obj#3,700) {this->ID3D12CommandQueue obj#1,return->S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	**Wait**(obj#4,83) {this->ID3D12CommandQueue obj#1,return->S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	**ResourceBarrier**(1,...) {this->ID3D12GraphicsCommandList obj#5}	0	365	573	938	0	0	0	0	0	0	0	0	0	0	0
10,571	AutomaticExposure	4,433,594	4,434,376	72,812	73,594	0	0	0	0	0				0	235,520	0
10,584	ToneMapping	4,507,188	4,506,980	81,823	81,615	0	0	0	0	0				0	3,686,400	0
10,644	**Signal**(obj#3,701) {this->ID3D12CommandQueue obj#1,return->S_OK}	4,679,011	4,679,011	0	0	0	0	0	0	0	0	0	0	0	0	0
10,647	**ResourceBarrier**(1,...) {this->ID3D12GraphicsCommandList obj#3139}	4,679,011	4,680,052	573	1,614	0	0	0	0	0	0	0	0	0	0	0
10,648	**ResolveQueryData**(obj#626,D3D12_QUERY_TYPE_TIMESTAMP,512,20,obj#3141,4096) {this->ID3D12Grap…	4,680,625	4,680,157	3,541	3,073	0	0	0	0	0	0	0	0	0	0	0
10,650	**Signal**(obj#3,702) {this->ID3D12CommandQueue obj#1,return->S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,651	**Signal**(obj#3142,86) {this->ID3D12CommandQueue obj#1,return->S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,652	**Present**(obj#7,0,<unknown>) {this->ID3D12SharingContract obj#1}					0	0	0	0	0	0	0	0	0	0	0
10,458	[redacted]	3,000,156	3,001,458	1,418,490	1,419,792	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	13,976,576	0
10,551	SkyDome	4,419,948	4,421,094	12,500	13,646	3	1	3	1	1	3,686,400	3,679,184	7,216	7,528	0	0
10,628	SwapChain	4,609,428	4,610,366	68,645	69,583	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	0	0
10,599	UI	4,588,803	4,591,615	17,813	20,625	8,358	2,786	5,222	2,786	2,786	1,141,812	0	1,141,812	1,245,076	0	0
15	RasterShadow	2,500	2,864	796,355	796,719	17,211,768	5,737,256	6,068,349	5,737,256	942,604	5,871,698	1,387,629	4,484,069	0	0	50
5,234	GBuffer	799,219	800,260	2,199,896	2,200,937	17,211,768	5,737,256	6,068,349	5,737,256	3,252,282	18,565,295	9,064,077	9,501,218	12,553,780	0	40

Counters

Pipeline counters

Filter (Ctrl+E)

- INTEL: Color Pipe
- INTEL: Copy Engine
- INTEL: Depth Pipe
- INTEL: Device Cache
- INTEL: Front End
- INTEL: GPU
- INTEL: Geometry — 4 selected
  - ☐ Clipper Input Available
  - ☑ Clipper Input Vertex
  - ☐ Clipper Output Ready
  - ☑ Clipper Primitive Cull
  - ☑ Clipper Primitive Far Near Clip
  - ☑ Clipper Primitive Output
  - ☐ Clipper Transaction Output
  - ☐ IA Primitive
  - ☐ IA Vertex
  - ☐ Streamout Output Vertex Count
  - ☐ Streamout Primitive Storage Need
  - ☐ Streamout Primitive Write Count
  - ☐ Stripsfan Object Count
  - ☐ Stripsfan Object Cull
  - ☐ Stripsfan Output Ready
  - ☐ VS Output Ready
  - ☐ Vertex Fetch Input Available
  - ☐ Vertex Fetch Output Ready

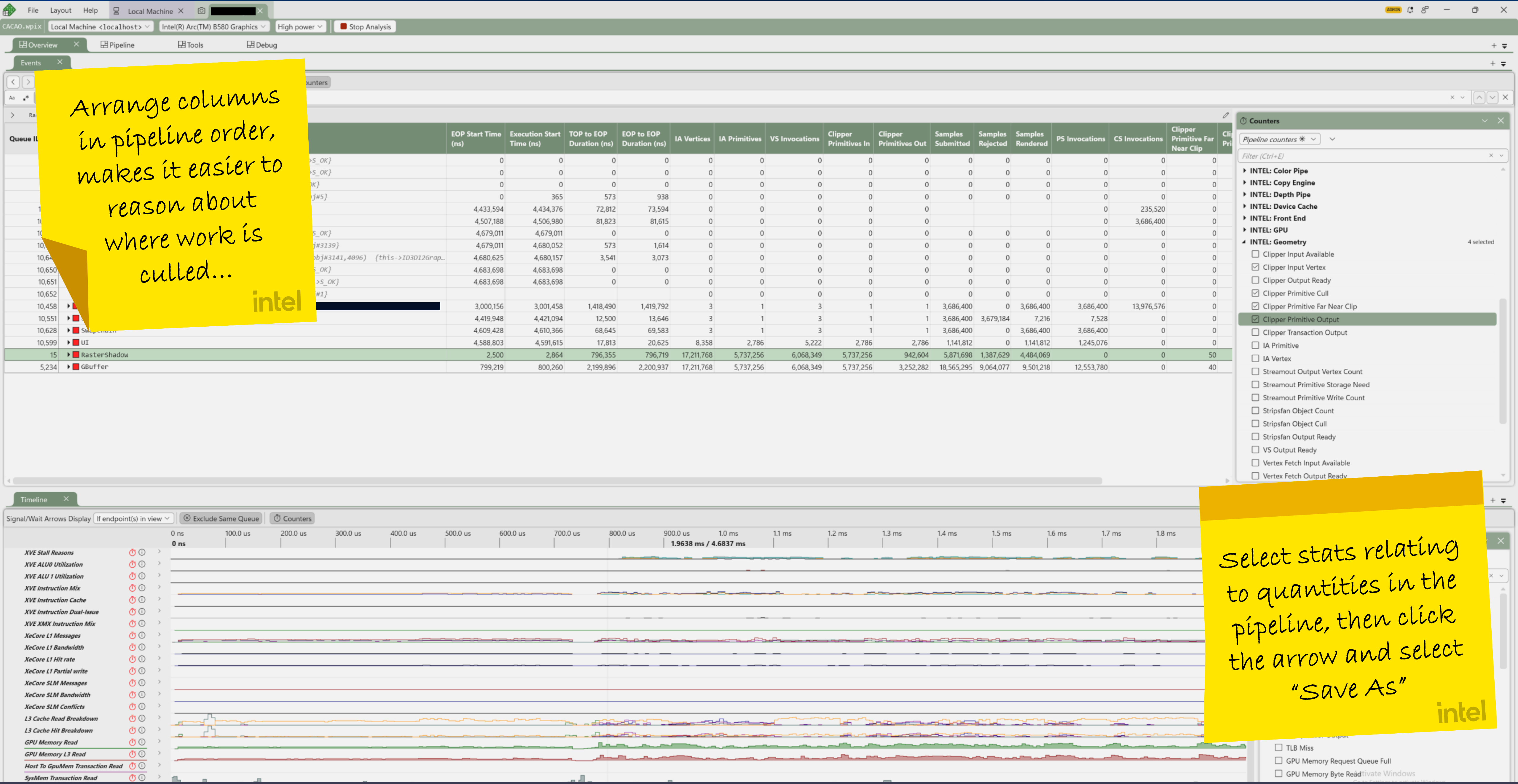
- ☐ TLB Miss
- ☐ GPU Memory Request Queue Full
- ☐ GPU Memory Byte Read

Select stats relating to quantities in the pipeline, then click the arrow and select "Save As"

intel

Arrange columns
in pipeline order,
makes it easier to
reason about
where work is
culled…

intel



Queue ID		EOP Start Time (ns)	Execution Start Time (ns)	TOP to EOP Duration (ns)	EOP to EOP Duration (ns)	IA Vertices	IA Primitives	VS Invocations	Clipper Primitives In	Clipper Primitives Out	Samples Submitted	Samples Rejected	Samples Rendered	PS Invocations	CS Invocations	Clipper Primitive Far Near Clip
	>S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	>S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	j#5}	0	365	573	938	0	0	0	0	0	0	0	0	0	0	0
1		4,433,594	4,434,376	72,812	73,594	0	0	0	0	0				0	235,520	0
10		4,507,188	4,506,980	81,823	81,615	0	0	0	0	0				0	3,686,400	0
10	S_OK}	4,679,011	4,679,011	0	0	0	0	0	0	0	0	0	0	0	0	0
10,	j#3139}	4,679,011	4,680,052	573	1,614	0	0	0	0	0	0	0	0	0	0	0
10,64	obj#3141,4096) {this->ID3D12Grap…	4,680,625	4,680,157	3,541	3,073	0	0	0	0	0	0	0	0	0	0	0
10,650	S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,651	>S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,652	#1}					0	0	0	0	0	0	0	0	0	0	0
10,458	[redacted]	3,000,156	3,001,458	1,418,490	1,419,792	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	13,976,576	0
10,551		4,419,948	4,421,094	12,500	13,646	3	1	3	1	1	3,686,400	3,679,184	7,216	7,528	0	0
10,628	Swapchain	4,609,428	4,610,366	68,645	69,583	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	0	0
10,599	UI	4,588,803	4,591,615	17,813	20,625	8,358	2,786	5,222	2,786	2,786	1,141,812	0	1,141,812	1,245,076	0	0
15	RasterShadow	2,500	2,864	796,355	796,719	17,211,768	5,737,256	6,068,349	5,737,256	942,604	5,871,698	1,387,629	4,484,069	0	0	50
5,234	GBuffer	799,219	800,260	2,199,896	2,200,937	17,211,768	5,737,256	6,068,349	5,737,256	3,252,282	18,565,295	9,064,077	9,501,218	12,553,780	0	40

Counters

Pipeline counters ✳

Filter (Ctrl+E)

- INTEL: Color Pipe
- INTEL: Copy Engine
- INTEL: Depth Pipe
- INTEL: Device Cache
- INTEL: Front End
- INTEL: GPU
- INTEL: Geometry — 4 selected
  - ☐ Clipper Input Available
  - ☑ Clipper Input Vertex
  - ☐ Clipper Output Ready
  - ☑ Clipper Primitive Cull
  - ☑ Clipper Primitive Far Near Clip
  - ☑ Clipper Primitive Output
  - ☐ Clipper Transaction Output
  - ☐ IA Primitive
  - ☐ IA Vertex
  - ☐ Streamout Output Vertex Count
  - ☐ Streamout Primitive Storage Need
  - ☐ Streamout Primitive Write Count
  - ☐ Stripsfan Object Count
  - ☐ Stripsfan Object Cull
  - ☐ Stripsfan Output Ready
  - ☐ VS Output Ready
  - ☐ Vertex Fetch Input Available
  - ☐ Vertex Fetch Output Ready

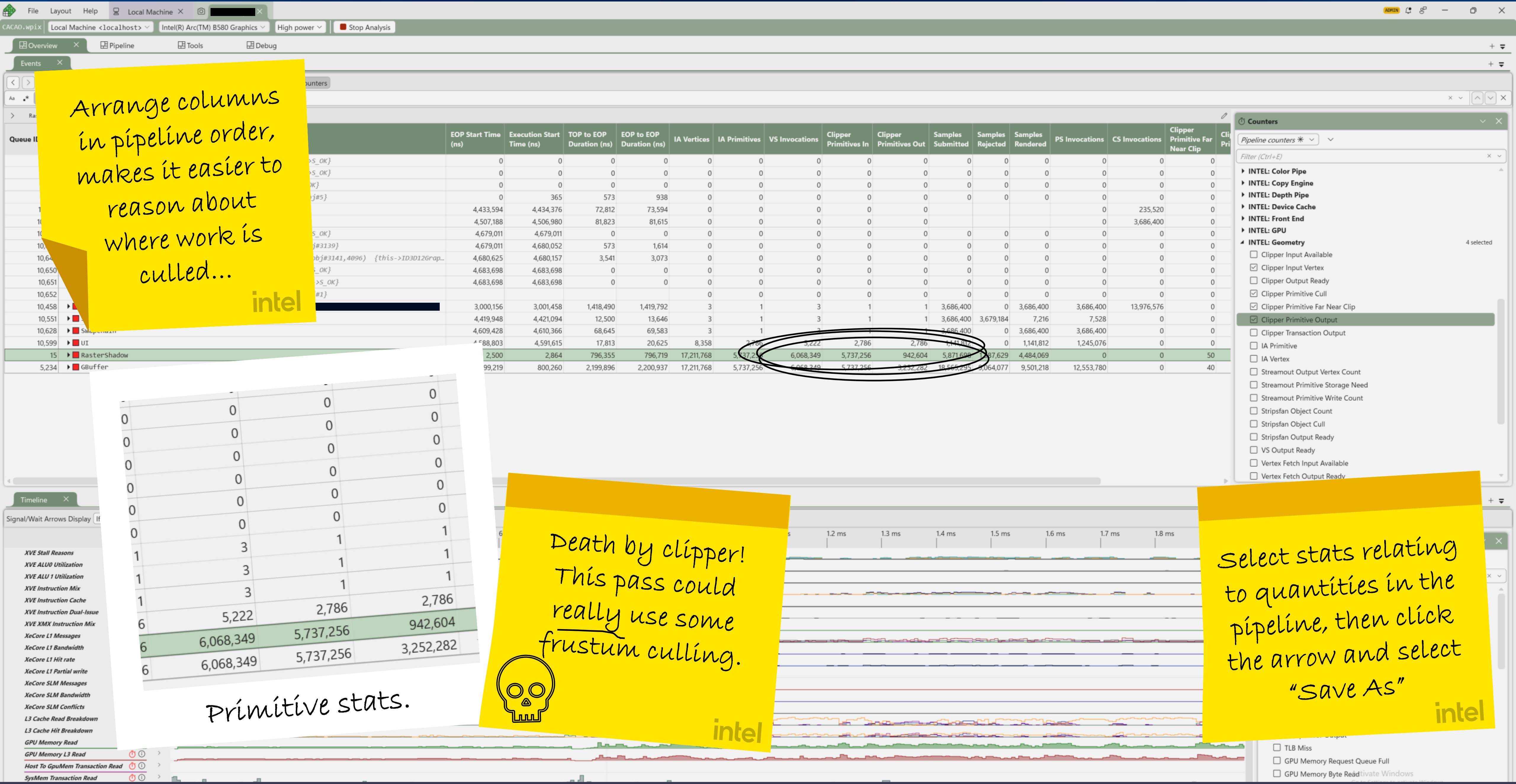
☐ TLB Miss

☐ GPU Memory Request Queue Full

☐ GPU Memory Byte Read

Select stats relating
to quantities in the
pipeline, then click
the arrow and select
"Save As"

intel



Arrange columns in pipeline order, makes it easier to reason about where work is culled…

Queue ID		EOP Start Time (ns)	Execution Start Time (ns)	TOP to EOP Duration (ns)	EOP to EOP Duration (ns)	IA Vertices	IA Primitives	VS Invocations	Clipper Primitives In	Clipper Primitives Out	Samples Submitted	Samples Rejected	Samples Rendered	PS Invocations	CS Invocations	Clipper Primitive Far Near Clip
	>S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	>S_OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OK}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	j#5}	0	365	573	938	0	0	0	0	0	0	0	0	0	0	0
1		4,433,594	4,434,376	72,812	73,594	0	0	0	0	0				0	235,520	0
1		4,507,188	4,506,980	81,823	81,615	0	0	0	0	0				0	3,686,400	0
1	S_OK}	4,679,011	4,679,011	0	0	0	0	0	0	0	0	0	0	0	0	0
10,	j#3139}	4,679,011	4,680,052	573	1,614	0	0	0	0	0	0	0	0	0	0	0
10,64	obj#3141,4096) {this->ID3D12Grap…	4,680,625	4,680,157	3,541	3,073	0	0	0	0	0	0	0	0	0	0	0
10,650	S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,651	>S_OK}	4,683,698	4,683,698	0	0	0	0	0	0	0	0	0	0	0	0	0
10,652	#1}					0	0	0	0	0	0	0	0	0	0	0
10,458	[redacted]	3,000,156	3,001,458	1,418,490	1,419,792	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	13,976,576	0
10,551		4,419,948	4,421,094	12,500	13,646	3	1	3	1	1	3,686,400	3,679,184	7,216	7,528	0	0
10,628	Sw	4,609,428	4,610,366	68,645	69,583	3	1	3	1	1	3,686,400	0	3,686,400	3,686,400	0	0
10,599	UI	4,588,803	4,591,615	17,813	20,625	8,358	2,786	5,222	2,786	2,786	1,141,812	0	1,141,812	1,245,076	0	0
15	RasterShadow	2,500	2,864	796,355	796,719	17,211,768	5,737,256	6,068,349	5,737,256	942,604	5,871,698	1,387,629	4,484,069	0	0	50
5,234	GBuffer	99,219	800,260	2,199,896	2,200,937	17,211,768	5,737,256	6,068,349	5,737,256	3,252,282	18,565,295	9,064,077	9,501,218	12,553,780	0	40

Counters

Pipeline counters ✳

Filter (Ctrl+E)

- INTEL: Color Pipe
- INTEL: Copy Engine
- INTEL: Depth Pipe
- INTEL: Device Cache
- INTEL: Front End
- INTEL: GPU
- INTEL: Geometry — 4 selected
  - ☐ Clipper Input Available
  - ☑ Clipper Input Vertex
  - ☐ Clipper Output Ready
  - ☑ Clipper Primitive Cull
  - ☑ Clipper Primitive Far Near Clip
  - ☑ Clipper Primitive Output
  - ☐ Clipper Transaction Output
  - ☐ IA Primitive
  - ☐ IA Vertex
  - ☐ Streamout Output Vertex Count
  - ☐ Streamout Primitive Storage Need
  - ☐ Streamout Primitive Write Count
  - ☐ Stripsfan Object Count
  - ☐ Stripsfan Object Cull
  - ☐ Stripsfan Output Ready
  - ☐ VS Output Ready
  - ☐ Vertex Fetch Input Available
  - ☐ Vertex Fetch Output Ready

- ☐ TLB Miss
- ☐ GPU Memory Request Queue Full
- ☐ GPU Memory Byte Read

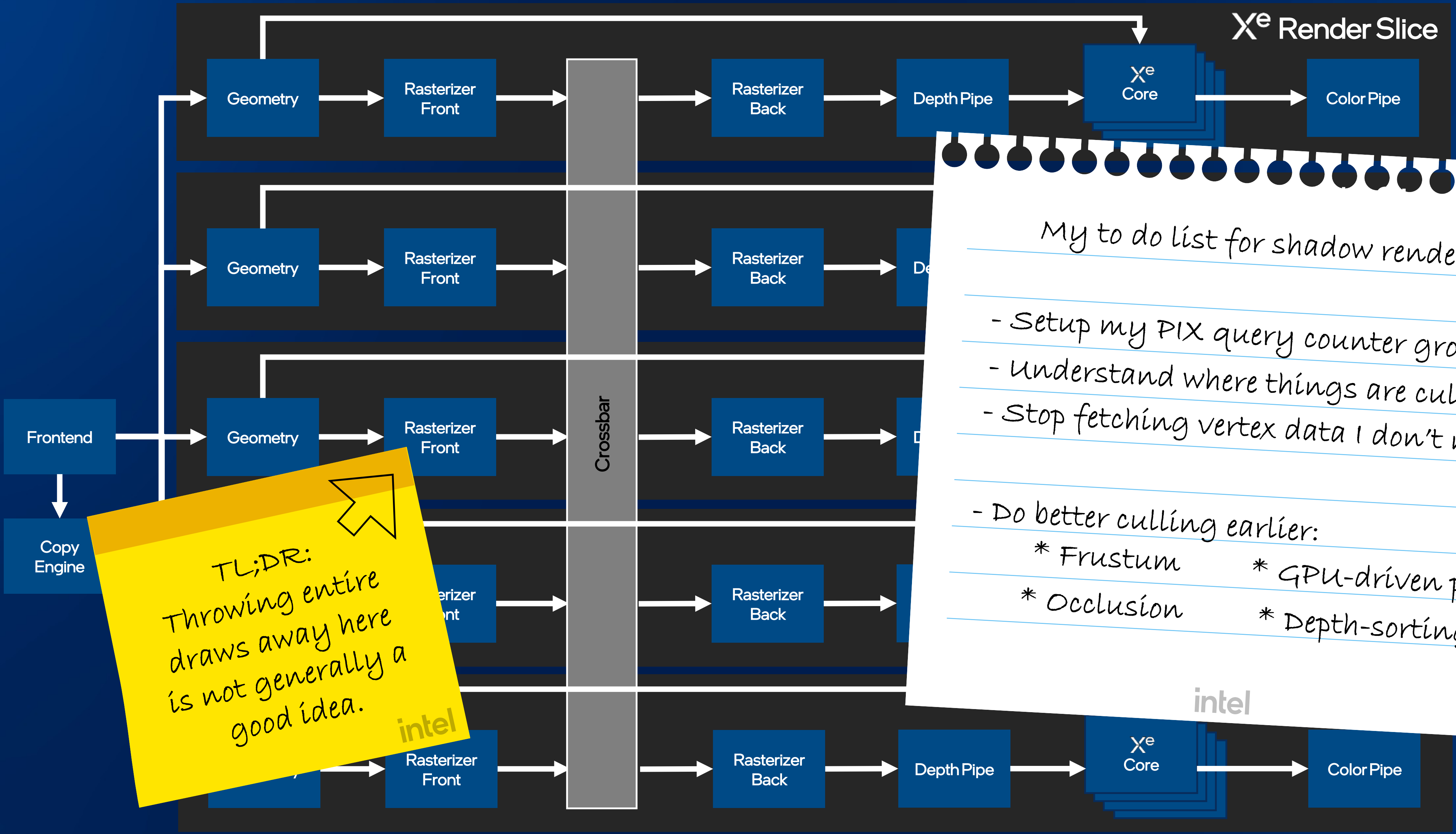
XVE Stall Reasons
XVE ALU0 Utilization
XVE ALU 1 Utilization
XVE Instruction Mix
XVE Instruction Cache
XVE Instruction Dual-Issue
XVE XMX Instruction Mix
XeCore L1 Messages
XeCore L1 Bandwidth
XeCore L1 Hit rate
XeCore L1 Partial write
XeCore SLM Messages
XeCore SLM Bandwidth
XeCore SLM Conflicts
L3 Cache Read Breakdown
L3 Cache Hit Breakdown
GPU Memory Read
GPU Memory L3 Read
Host To GpuMem Transaction Read
SysMem Transaction Read

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
1	3	1	1
1	3	1	1
1	3	1	1
6	5,222	2,786	2,786
6	6,068,349	5,737,256	942,604
6	6,068,349	5,737,256	3,252,282

Primitive stats.

Death by clipper! This pass could really use some frustum culling.

Select stats relating to quantities in the pipeline, then click pipeline, then click the arrow and select "Save As"



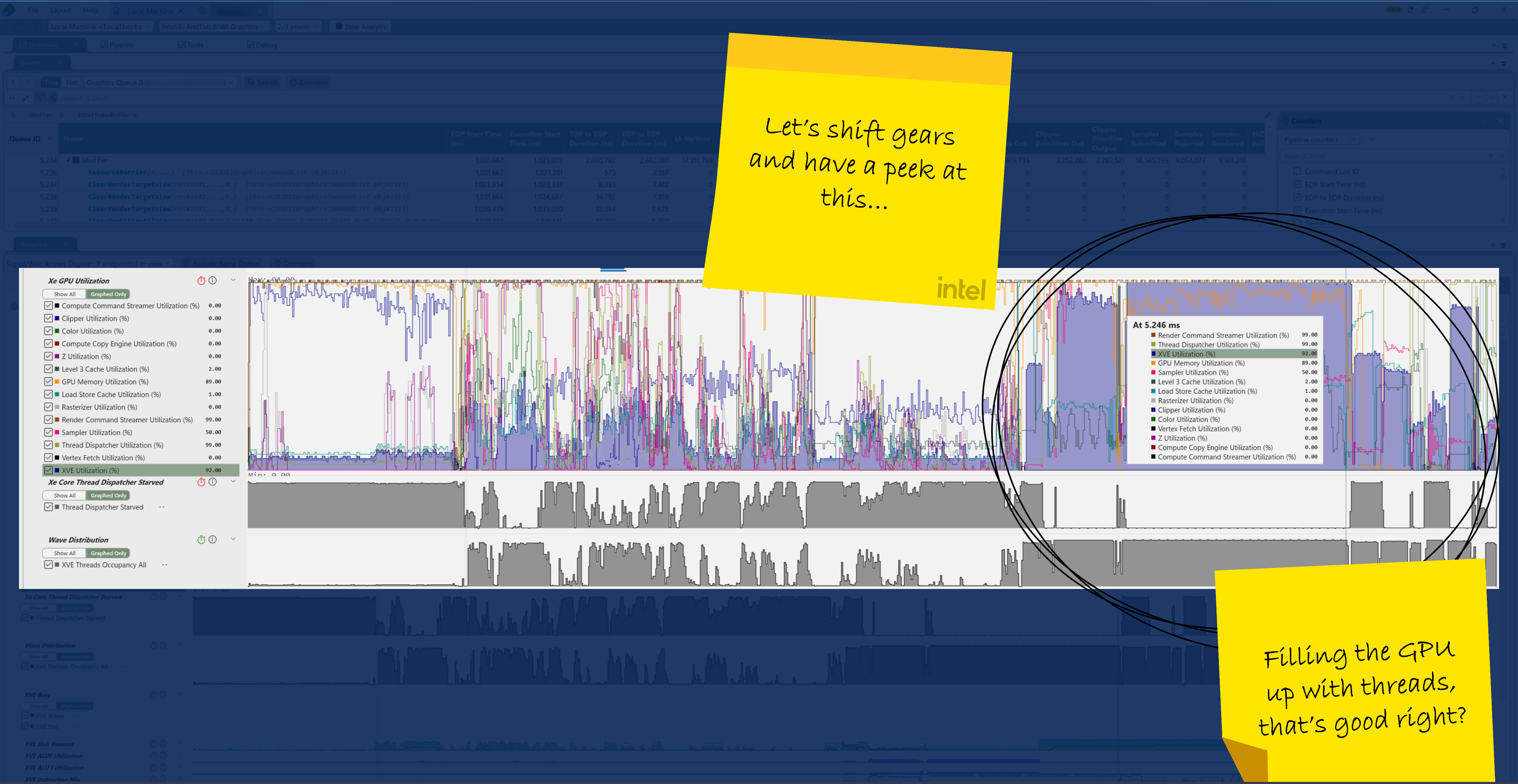
TL;DR:
Throwing entire draws away here is not generally a good idea.

My to do list for shadow rendering

- Setup my PIX query counter groups
- Understand where things are culled
- Stop fetching vertex data I don't need

- Do better culling earlier:
  * Frustum
  * GPU-driven pipe.
  * Occlusion
  * Depth-sorting

Let's shift gears and have a peek at this...



Xe GPU Utilization

Show All | Graphed Only

☑ Compute Command Streamer Utilization (%)	0.00
☑ Clipper Utilization (%)	0.00
☑ Color Utilization (%)	0.00
☑ Compute Copy Engine Utilization (%)	0.00
☑ Z Utilization (%)	0.00
☑ Level 3 Cache Utilization (%)	2.00
☑ GPU Memory Utilization (%)	89.00
☑ Load Store Cache Utilization (%)	1.00
☑ Rasterizer Utilization (%)	0.00
☑ Render Command Streamer Utilization (%)	99.00
☑ Sampler Utilization (%)	50.00
☑ Thread Dispatcher Utilization (%)	99.00
☑ Vertex Fetch Utilization (%)	0.00
☑ XVE Utilization (%)	92.00

Xe Core Thread Dispatcher Starved

Show All | Graphed Only

☑ Thread Dispatcher Starved --

Wave Distribution

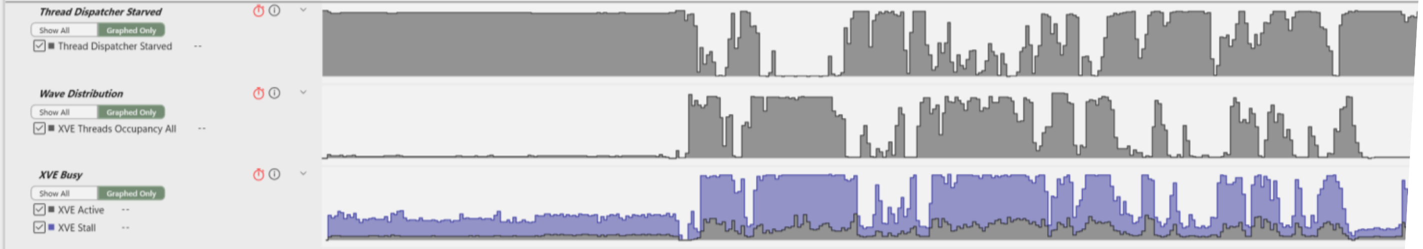
Show All | Graphed Only

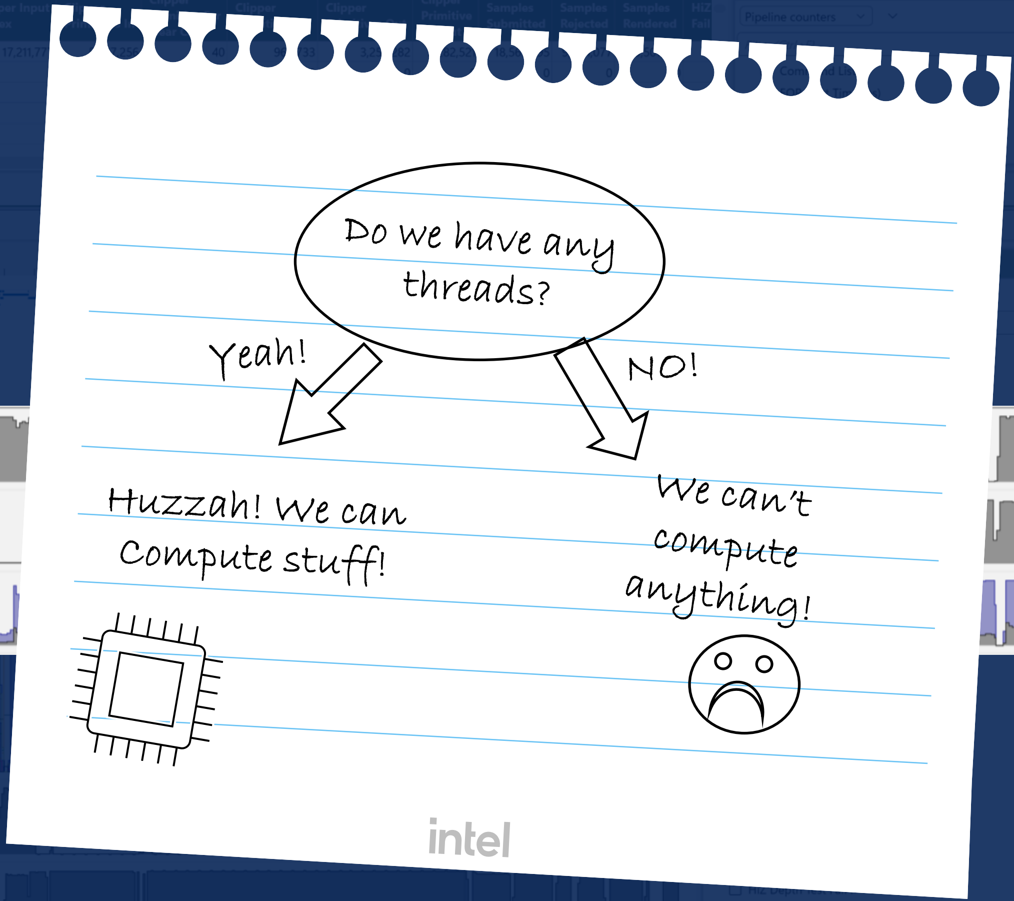
☑ XVE Threads Occupancy All --

At 5.246 ms

Render Command Streamer Utilization (%)	99.00
Thread Dispatcher Utilization (%)	99.00
XVE Utilization (%)	92.00
GPU Memory Utilization (%)	89.00
Sampler Utilization (%)	50.00
Level 3 Cache Utilization (%)	2.00
Load Store Cache Utilization (%)	1.00
Rasterizer Utilization (%)	0.00
Clipper Utilization (%)	0.00
Color Utilization (%)	0.00
Vertex Fetch Utilization (%)	0.00
Z Utilization (%)	0.00
Compute Copy Engine Utilization (%)	0.00
Compute Command Streamer Utilization (%)	0.00

Filling the GPU up with threads, that's good right?




Do we have any threads?

Yeah!

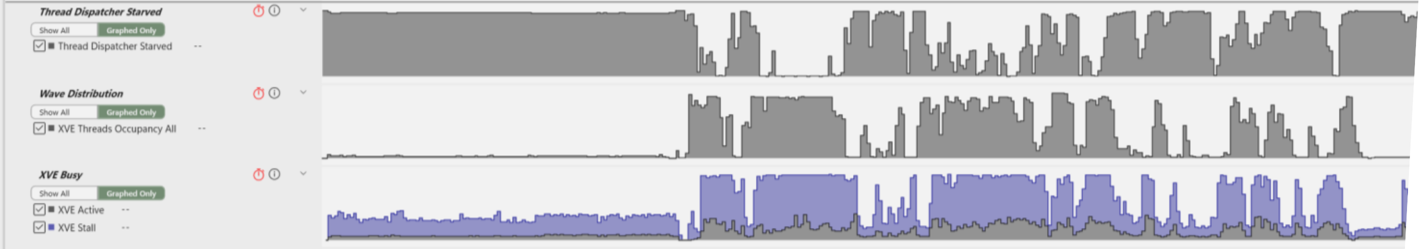
NO!

Huzzah! We can compute stuff!

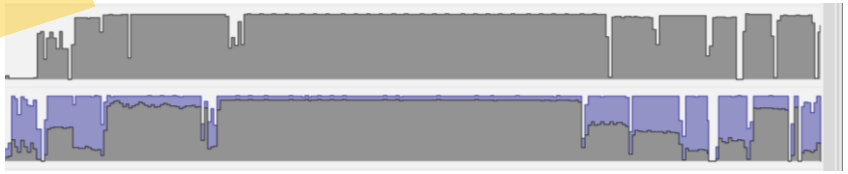
We can't compute anything!

XVE Stalled
+
XVE Active
=
XVE Busy

…but are more threads always better?

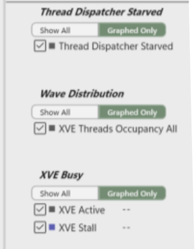


XVE Thread Occupancy



XVE busy

intel

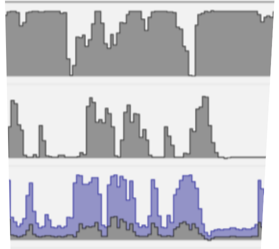


$\rho\ XVE_{active}, XVE_{occupancy}$

~0.132

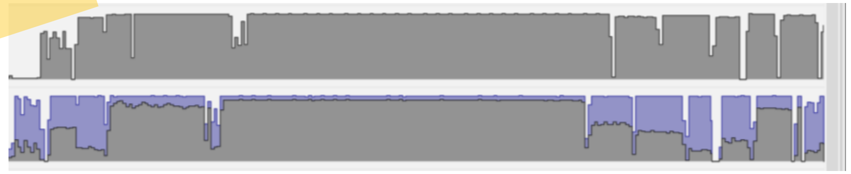
$\rho\ XVE_{busy}, XVE_{occupancy}$

~0.685



XVE Stalled
+
XVE Active
=
XVE Busy

...but are more threads *always* better?



Pearson of 0 is not correlated, 1 is highly correlated.

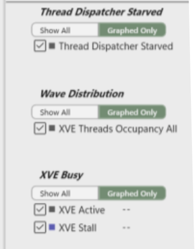
XVE Stalled
+
XVE Active
=
XVE Busy

$\rho \; XVE_{active} , XVE_{occupancy}$

~0.132

$\rho \; XVE_{busy} , XVE_{occupancy}$

~0.685


Thread Dispatcher Starved

Show All | Graphed Only

☑ ■ Thread Dispatcher Starved

Wave Distribution

Show All | Graphed Only

☑ ■ XVE Threads Occupancy All

XVE Busy

Show All | Graphed Only

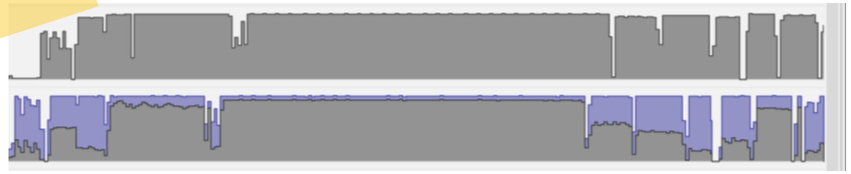
☑ ■ XVE Active --

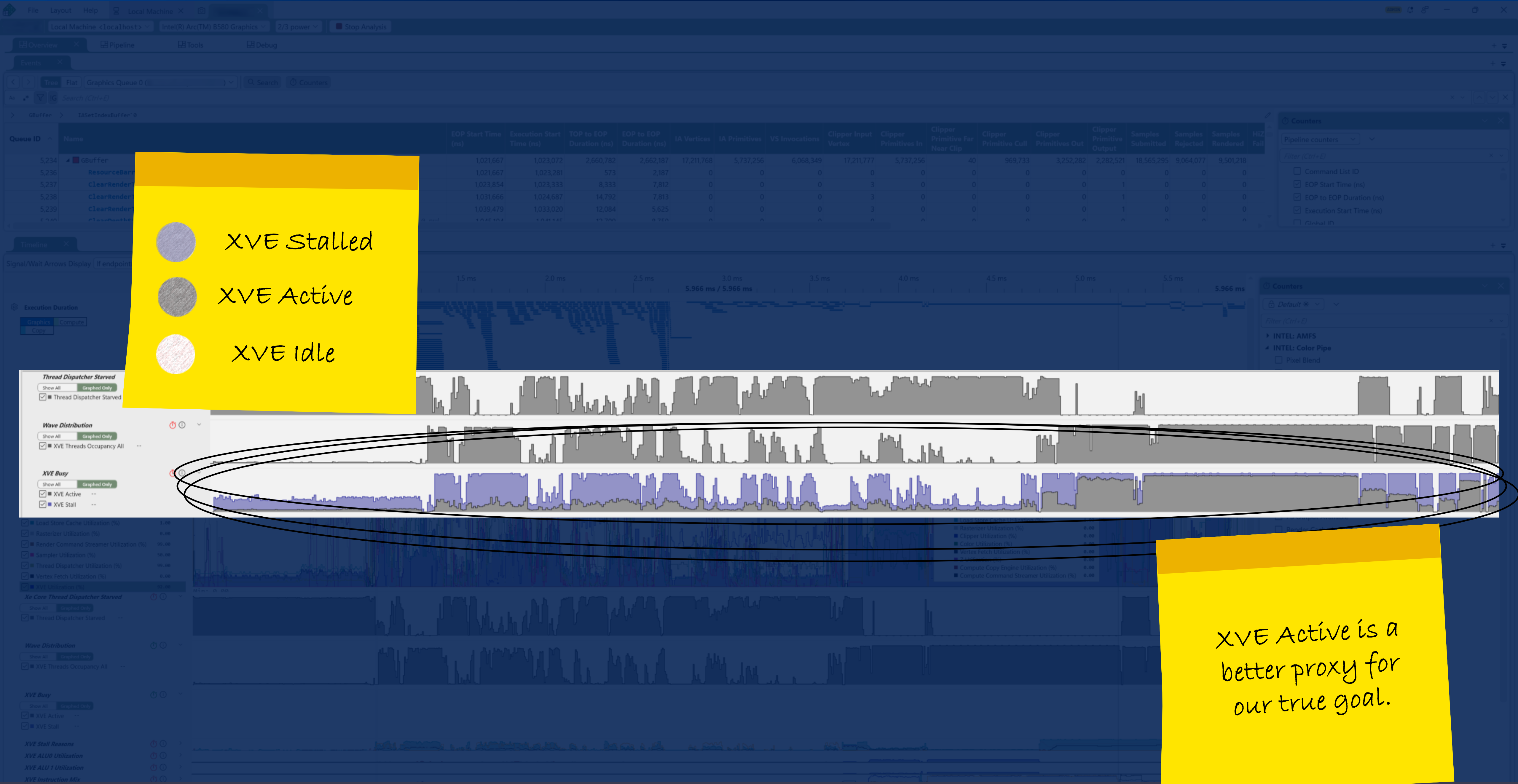
☑ ■ XVE Stall --

Pearson of 0 is not correlated, 1 is highly correlated.

Remember: Threads are just another tool to achieve your goal. Not the goal itself.

...but are more threads always better?



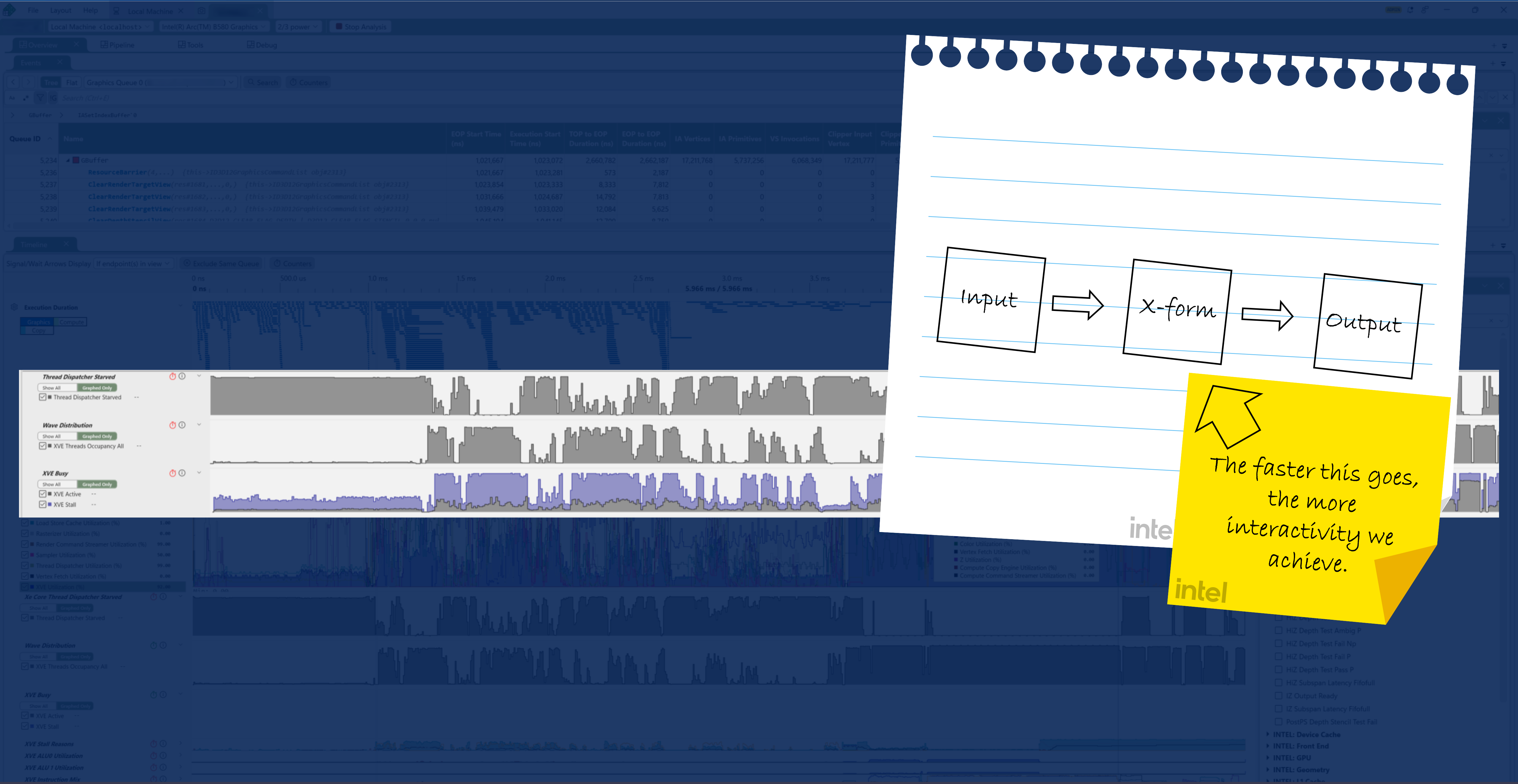


● XVE Stalled

● XVE Active

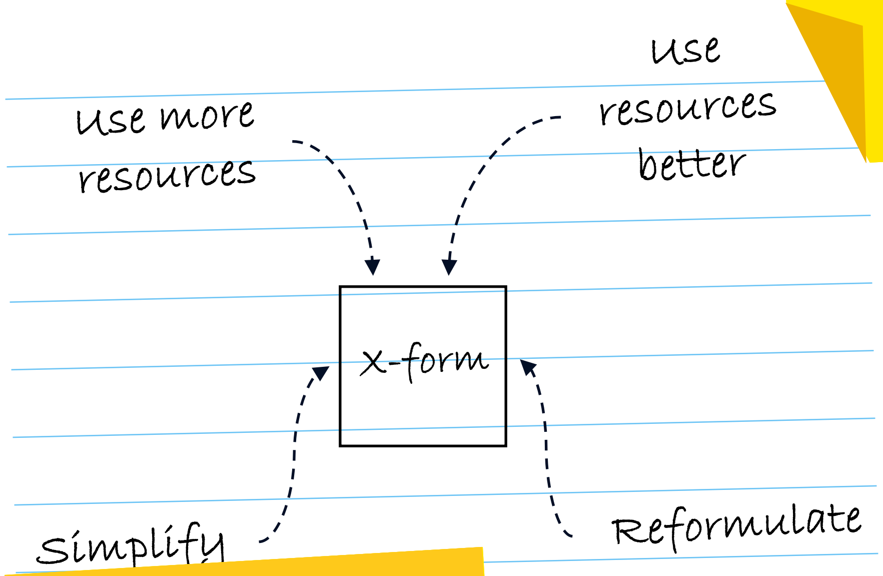
○ XVE Idle

XVE Active is a better proxy for our true goal.

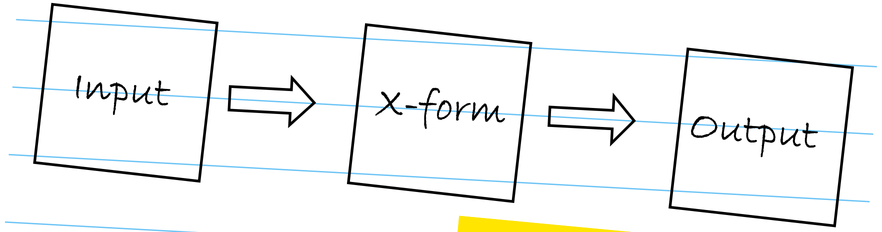


The faster this goes, the more interactivity we achieve.

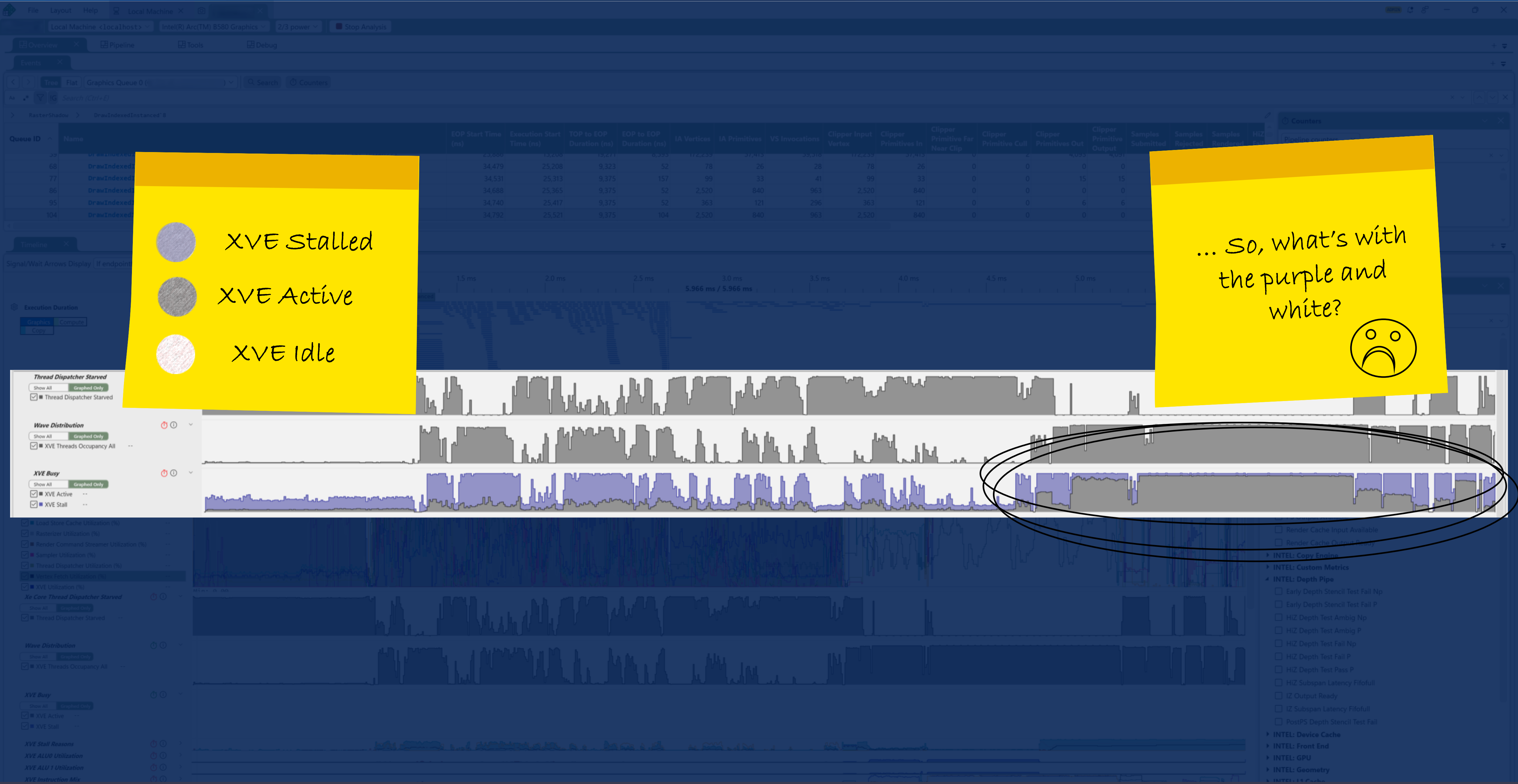
Resources are things like "compute horsepower" and "memory".



Using more available resources can speed up transformations.

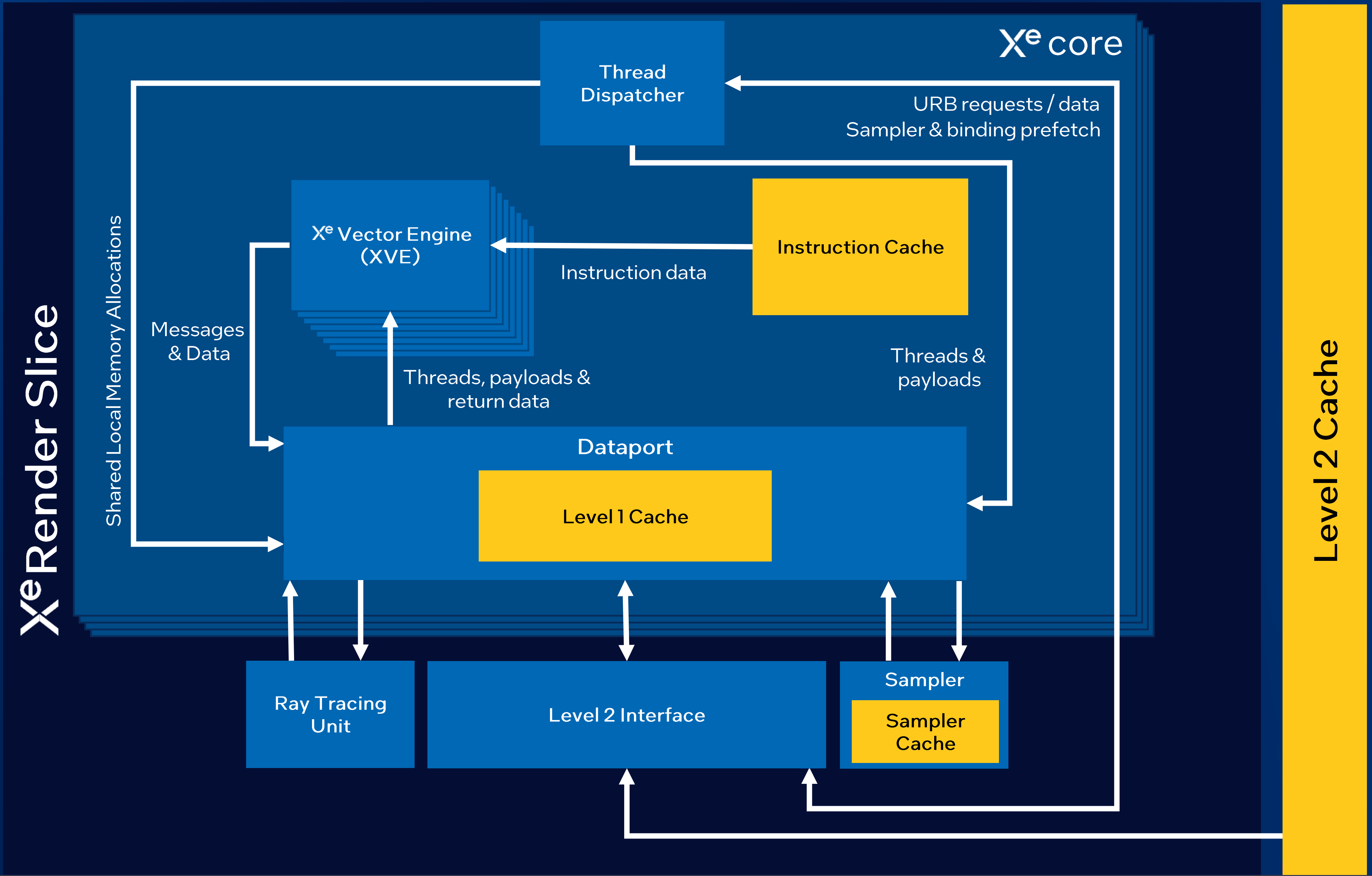


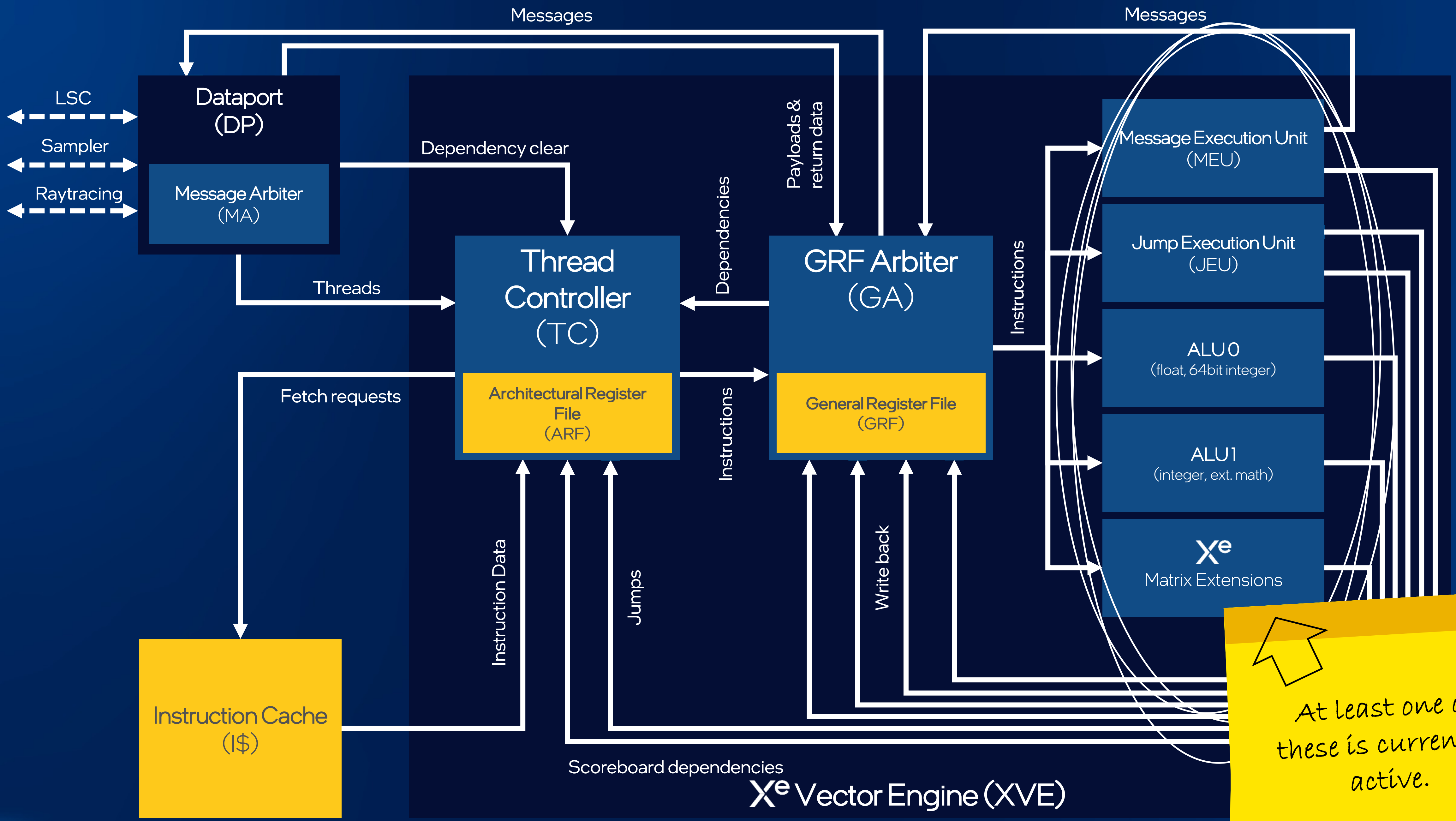
The faster this goes, the more interactivity we achieve.



● XVE Stalled
● XVE Active
○ XVE Idle

... So, what's with the purple and white?


X^e core
Thread Dispatcher
URB requests / data
Sampler & binding prefetch
X^e Vector Engine (XVE)
Instruction Cache
Instruction data
Messages & Data
Threads & payloads
Threads, payloads & return data
Dataport
Level 1 Cache
Shared Local Memory Allocations
X^e Render Slice
Ray Tracing Unit
Level 2 Interface
Sampler
Sampler Cache
Level 2 Cache



Messages

Messages

LSC

Sampler

Raytracing

Dataport
(DP)

Message Arbiter
(MA)

Dependency clear

Threads

Thread
Controller
(TC)

Architectural Register
File
(ARF)

Fetch requests

Dependencies

Payloads &
return data

Instructions

GRF Arbiter
(GA)

General Register File
(GRF)

Instructions

Message Execution Unit
(MEU)

Jump Execution Unit
(JEU)

ALU 0
(float, 64bit integer)

ALU 1
(integer, ext. math)

X^e
Matrix Extensions

Instruction Data

Jumps

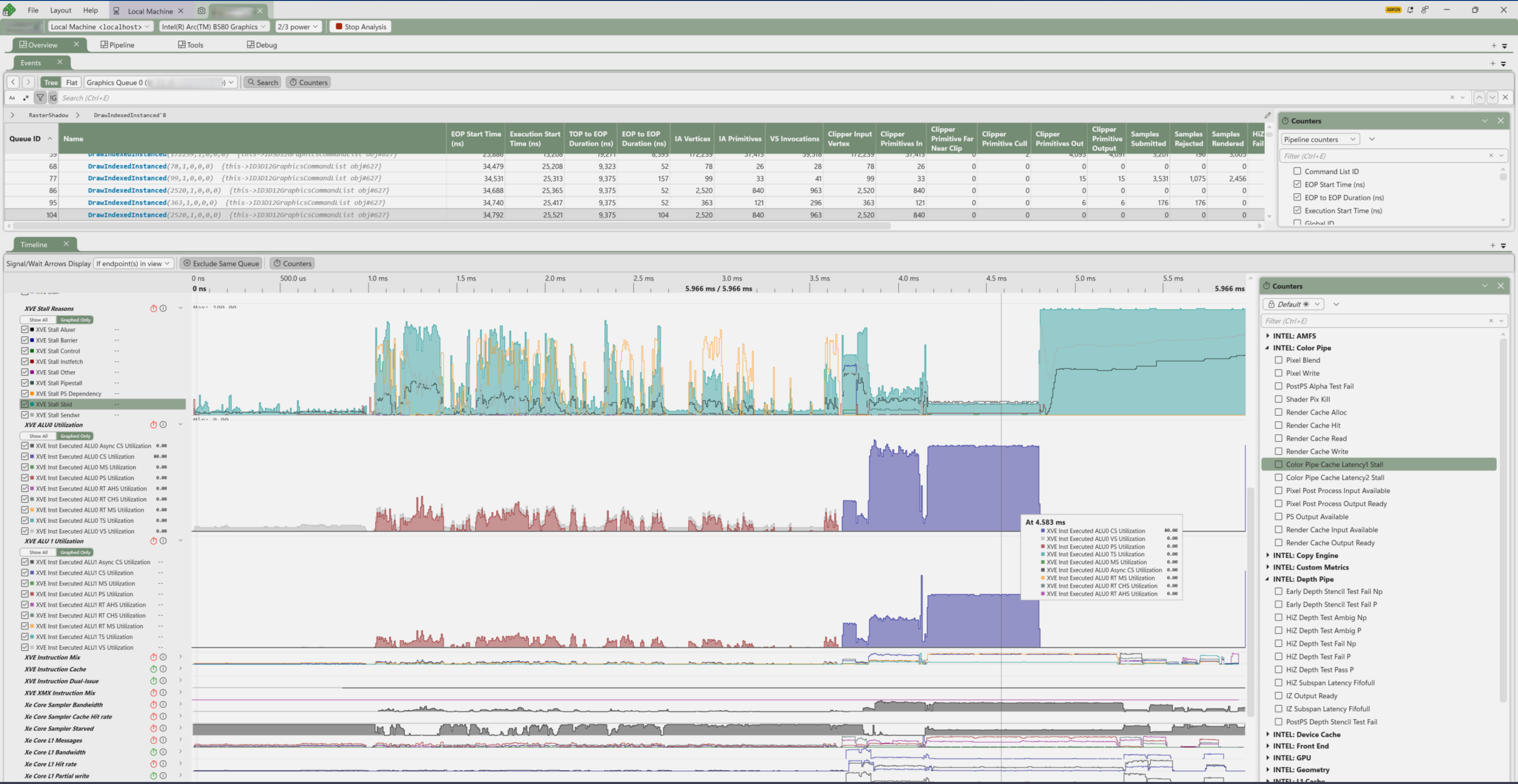
Write back

Instruction Cache
(I$)

Scoreboard dependencies

X^e Vector Engine (XVE)

At least one of
these is currently
active.



Queue ID	Name	EOP Start Time (ns)	Execution Start Time (ns)	TOP to EOP Duration (ns)	EOP to EOP Duration (ns)	IA Vertices	IA Primitives	VS Invocations	Clipper Input Vertex	Clipper Primitives In	Clipper Primitive Far Near Clip	Clipper Primitive Cull	Clipper Primitives Out	Clipper Primitive Output	Samples Submitted	Samples Rejected	Samples Rendered	HiZ Fail
68	DrawIndexedInstanced(78,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	34,479	25,208	9,323	52	78	26	28	78	26	0	0	0	0	0	0	0	
77	DrawIndexedInstanced(99,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	34,531	25,313	9,375	157	99	33	41	99	33	0	0	15	15	3,531	1,075	2,456	
86	DrawIndexedInstanced(2520,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	34,688	25,365	9,375	52	2,520	840	963	2,520	840	0	0	0	0	0	0	0	
95	DrawIndexedInstanced(363,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	34,740	25,417	9,375	52	363	121	296	363	121	0	0	6	6	176	176	0	
104	DrawIndexedInstanced(2520,1,0,0,0) {this->ID3D12GraphicsCommandList obj#627}	34,792	25,521	9,375	104	2,520	840	963	2,520	840	0	0	0	0	0	0	0	

Counters

Pipeline counters

Filter (Ctrl+E)

- ☐ Command List ID
- ☑ EOP Start Time (ns)
- ☑ EOP to EOP Duration (ns)
- ☑ Execution Start Time (ns)
- ☐ Global ID

XVE Stall Reasons

Show All | Graphed Only

- ☑ XVE Stall Aluwr --
- ☑ XVE Stall Barrier --
- ☑ XVE Stall Control --
- ☑ XVE Stall Instfetch --
- ☑ XVE Stall Other --
- ☑ XVE Stall Pipestall --
- ☑ XVE Stall PS Dependency --
- ☑ XVE Stall Sbid --
- ☑ XVE Stall Sendwr --

XVE ALU0 Utilization

Show All | Graphed Only

- ☑ XVE Inst Executed ALU0 Async CS Utilization 0.00
- ☑ XVE Inst Executed ALU0 CS Utilization 80.00
- ☑ XVE Inst Executed ALU0 MS Utilization 0.00
- ☑ XVE Inst Executed ALU0 PS Utilization 0.00
- ☑ XVE Inst Executed ALU0 RT AHS Utilization 0.00
- ☑ XVE Inst Executed ALU0 RT CHS Utilization 0.00
- ☑ XVE Inst Executed ALU0 RT MS Utilization 0.00
- ☑ XVE Inst Executed ALU0 TS Utilization 0.00
- ☑ XVE Inst Executed ALU0 VS Utilization 0.00

XVE ALU 1 Utilization

Show All | Graphed Only

- ☑ XVE Inst Executed ALU1 Async CS Utilization --
- ☑ XVE Inst Executed ALU1 CS Utilization --
- ☑ XVE Inst Executed ALU1 MS Utilization --
- ☑ XVE Inst Executed ALU1 PS Utilization --
- ☑ XVE Inst Executed ALU1 RT AHS Utilization --
- ☑ XVE Inst Executed ALU1 RT CHS Utilization --
- ☑ XVE Inst Executed ALU1 RT MS Utilization --
- ☑ XVE Inst Executed ALU1 TS Utilization --
- ☑ XVE Inst Executed ALU1 VS Utilization --

XVE Instruction Mix

XVE Instruction Cache

XVE Instruction Dual-Issue

XVE XMX Instruction Mix

Xe Core Sampler Bandwidth

Xe Core Sampler Cache Hit rate

Xe Core Sampler Starved

Xe Core L1 Messages

Xe Core L1 Bandwidth

Xe Core L1 Hit rate

Xe Core L1 Partial write

At 4.583 ms

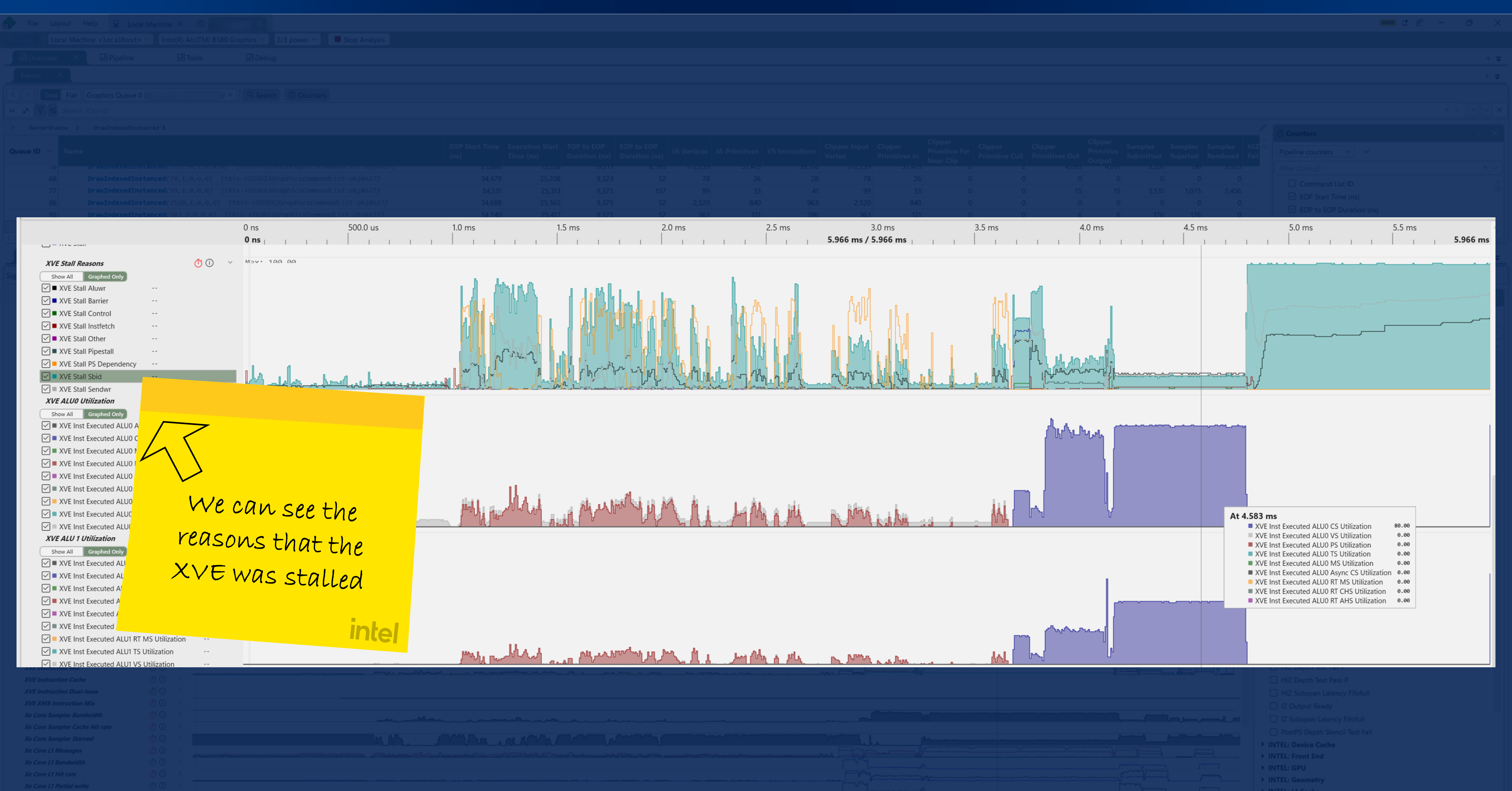
- XVE Inst Executed ALU0 CS Utilization 80.00
- XVE Inst Executed ALU0 VS Utilization 0.00
- XVE Inst Executed ALU0 PS Utilization 0.00
- XVE Inst Executed ALU0 TS Utilization 0.00
- XVE Inst Executed ALU0 MS Utilization 0.00
- XVE Inst Executed ALU0 Async CS Utilization 0.00
- XVE Inst Executed ALU0 RT MS Utilization 0.00
- XVE Inst Executed ALU0 RT CHS Utilization 0.00
- XVE Inst Executed ALU0 RT AHS Utilization 0.00

Counters

Default ✳

Filter (Ctrl+E)

- INTEL: AMFS
- INTEL: Color Pipe
  - ☐ Pixel Blend
  - ☐ Pixel Write
  - ☐ PostPS Alpha Test Fail
  - ☐ Shader Pix Kill
  - ☐ Render Cache Alloc
  - ☐ Render Cache Hit
  - ☐ Render Cache Read
  - ☐ Render Cache Write
  - ☐ Color Pipe Cache Latency1 Stall
  - ☐ Color Pipe Cache Latency2 Stall
  - ☐ Pixel Post Process Input Available
  - ☐ Pixel Post Process Output Ready
  - ☐ PS Output Available
  - ☐ Render Cache Input Available
  - ☐ Render Cache Output Ready
- INTEL: Copy Engine
- INTEL: Custom Metrics
- INTEL: Depth Pipe
  - ☐ Early Depth Stencil Test Fail Np
  - ☐ Early Depth Stencil Test Fail P
  - ☐ HiZ Depth Test Ambig Np
  - ☐ HiZ Depth Test Ambig P
  - ☐ HiZ Depth Test Fail Np
  - ☐ HiZ Depth Test Fail P
  - ☐ HiZ Depth Test Pass P
  - ☐ HiZ Subspan Latency Fifofull
  - ☐ IZ Output Ready
  - ☐ IZ Subspan Latency Fifofull
  - ☐ PostPS Depth Stencil Test Fail
- INTEL: Device Cache
- INTEL: Front End
- INTEL: GPU
- INTEL: Geometry



XVE Stall Reasons

- ☑ XVE Stall Aluwr --
- ☑ XVE Stall Barrier --
- ☑ XVE Stall Control --
- ☑ XVE Stall Instfetch --
- ☑ XVE Stall Other --
- ☑ XVE Stall Pipestall --
- ☑ XVE Stall PS Dependency --
- ☑ XVE Stall Sbid --
- ☑ XVE Stall Sendwr

XVE ALU0 Utilization

- ☑ XVE Inst Executed ALU0 A
- ☑ XVE Inst Executed ALU0 C
- ☑ XVE Inst Executed ALU0 M
- ☑ XVE Inst Executed ALU0
- ☑ XVE Inst Executed ALU0
- ☑ XVE Inst Executed ALU0
- ☑ XVE Inst Executed ALU0
- ☑ XVE Inst Executed ALU0
- ☑ XVE Inst Executed ALU0

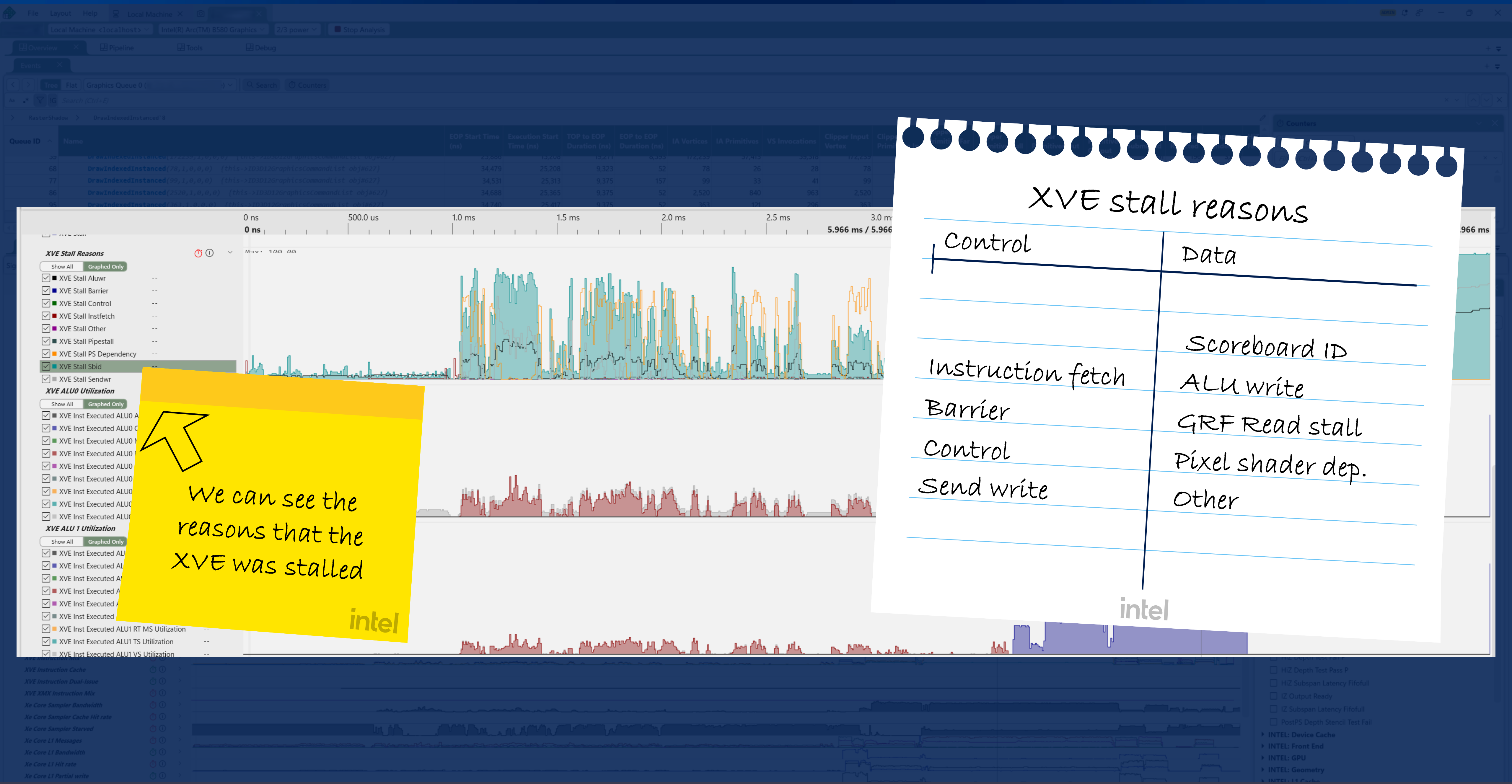
XVE ALU 1 Utilization

- ☑ XVE Inst Executed AL
- ☑ XVE Inst Executed AL
- ☑ XVE Inst Executed A
- ☑ XVE Inst Executed A
- ☑ XVE Inst Executed A
- ☑ XVE Inst Executed
- ☑ XVE Inst Executed ALU1 RT MS Utilization --
- ☑ XVE Inst Executed ALU1 TS Utilization --
- ☑ XVE Inst Executed ALU1 VS Utilization --

At 4.583 ms

XVE Inst Executed ALU0 CS Utilization	80.00
XVE Inst Executed ALU0 VS Utilization	0.00
XVE Inst Executed ALU0 PS Utilization	0.00
XVE Inst Executed ALU0 TS Utilization	0.00
XVE Inst Executed ALU0 MS Utilization	0.00
XVE Inst Executed ALU0 Async CS Utilization	0.00
XVE Inst Executed ALU0 RT MS Utilization	0.00
XVE Inst Executed ALU0 RT CHS Utilization	0.00
XVE Inst Executed ALU0 RT AHS Utilization	0.00

We can see the reasons that the XVE was stalled

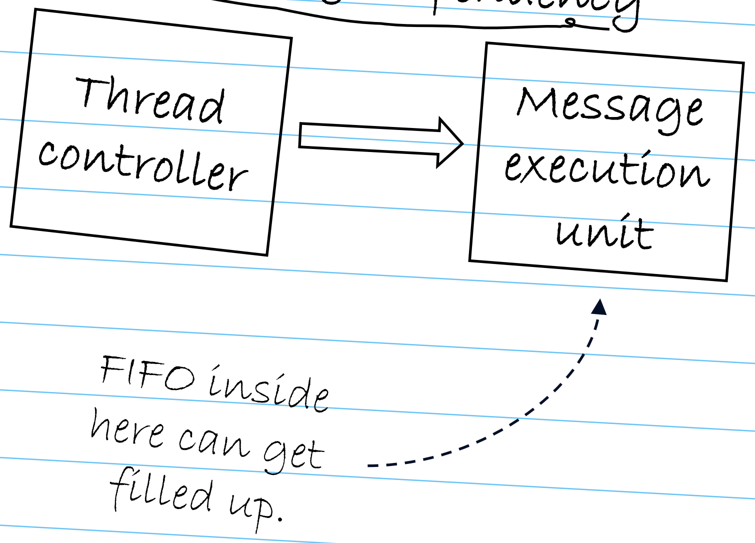


We can see the reasons that the XVE was stalled

XVE stall reasons

Control	Data
	Scoreboard ID
Instruction fetch	ALU write
Barrier	GRF Read stall
Control	Pixel shader dep.
Send write	Other

Message dependency



Thread controller

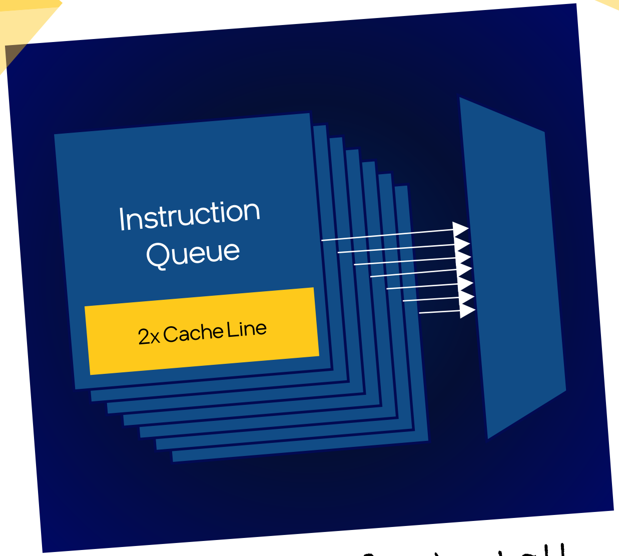
Message execution unit

FIFO inside here can get filled up.

Messages take time to be handled by their target shared function.

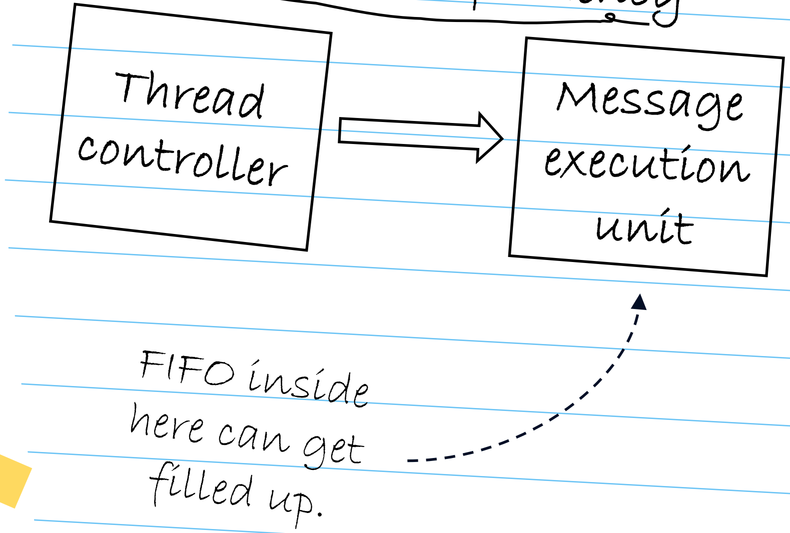
Instruction fetch stall

If there are no instruction cache lines available for a thread, then we can't run any code for that thread.



Instruction fetch stall

Message dependency

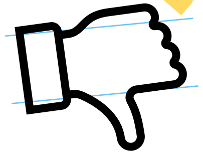


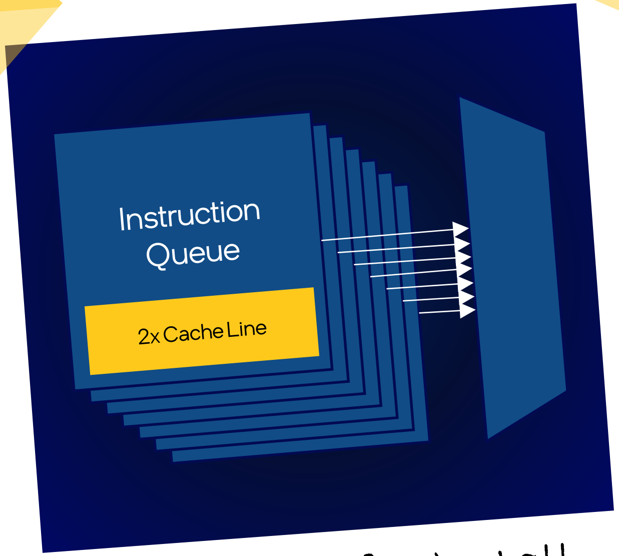
FIFO inside here can get filled up.

Messages take time to be handled by their target shared function.

Instruction fetch stall

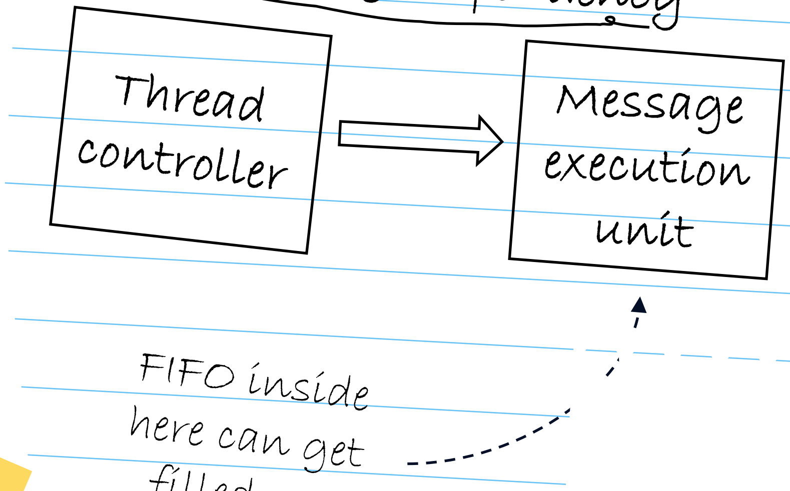
If there are no instruction cache lines available for a thread, then we can't run any code for that thread.





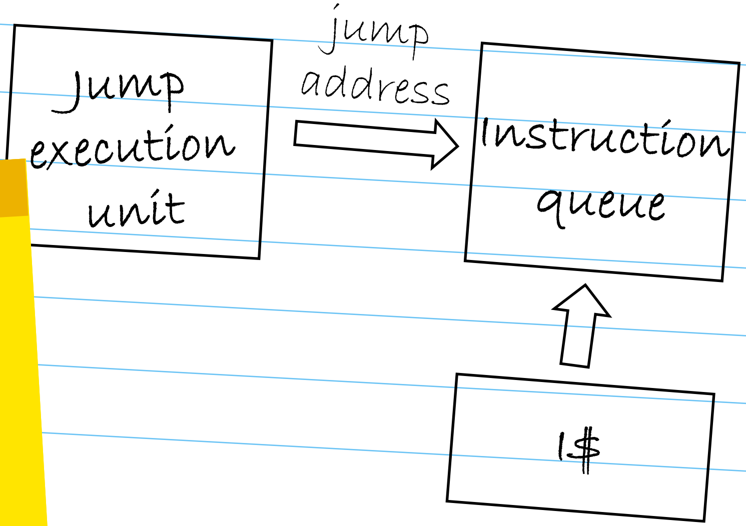
Instruction fetch stall

Message dependency

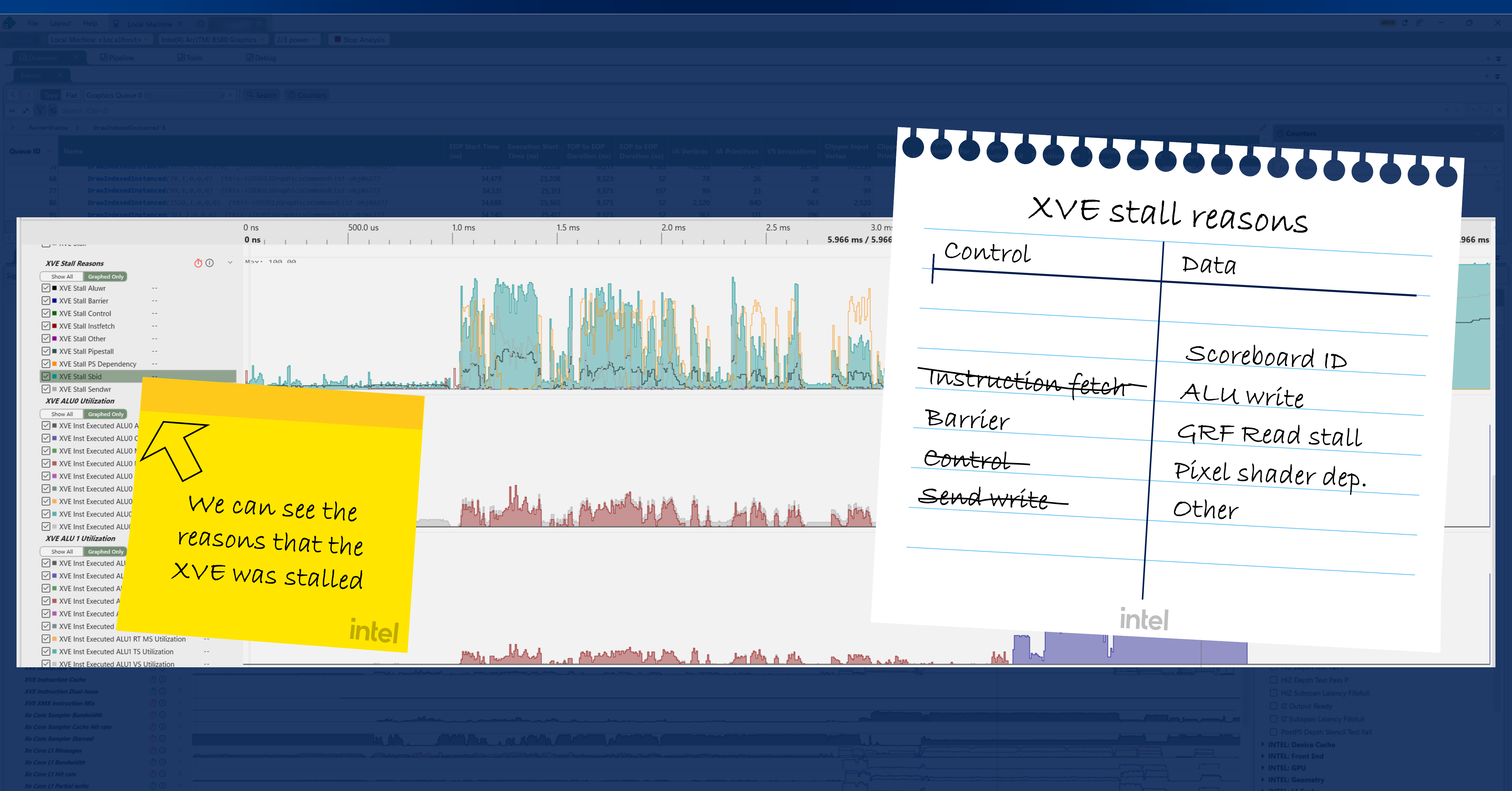


Messages take time to be handled by their target shared function.

Jump dependency



When we don't yet know the new instruction pointer, we can't execute code.



XVE Stall Reasons

Show All | Graphed Only

- ☑ XVE Stall Aluwr --
- ☑ XVE Stall Barrier --
- ☑ XVE Stall Control --
- ☑ XVE Stall Instfetch --
- ☑ XVE Stall Other --
- ☑ XVE Stall Pipestall --
- ☑ XVE Stall PS Dependency --
- ☑ XVE Stall Sbid --
- ☑ XVE Stall Sendwr --

XVE ALU0 Utilization

XVE ALU 1 Utilization

- ☑ XVE Inst Executed ALU1 RT MS Utilization --
- ☑ XVE Inst Executed ALU1 TS Utilization --
- ☑ XVE Inst Executed ALU1 VS Utilization --

We can see the reasons that the XVE was stalled

intel

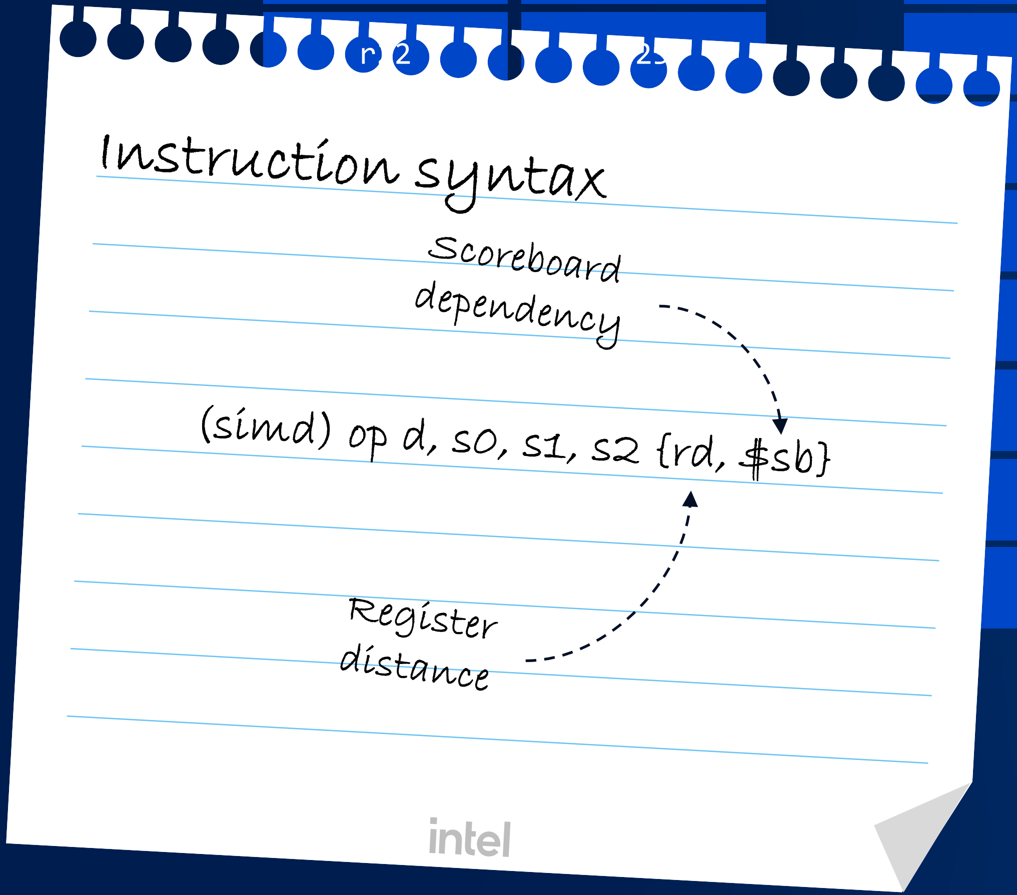
XVE stall reasons

Control	Data
	Scoreboard ID
~~Instruction fetch~~	ALU write
Barrier	GRF Read stall
~~Control~~	Pixel shader dep.
~~Send write~~	Other

intel

Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2			-	-
0x00000020	add	r6			1	-
0x00000030	add	r8			-	-
0x00000040	send	r10			2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6			-	$0.src
0x00000070	add	r12			2	$0.dst

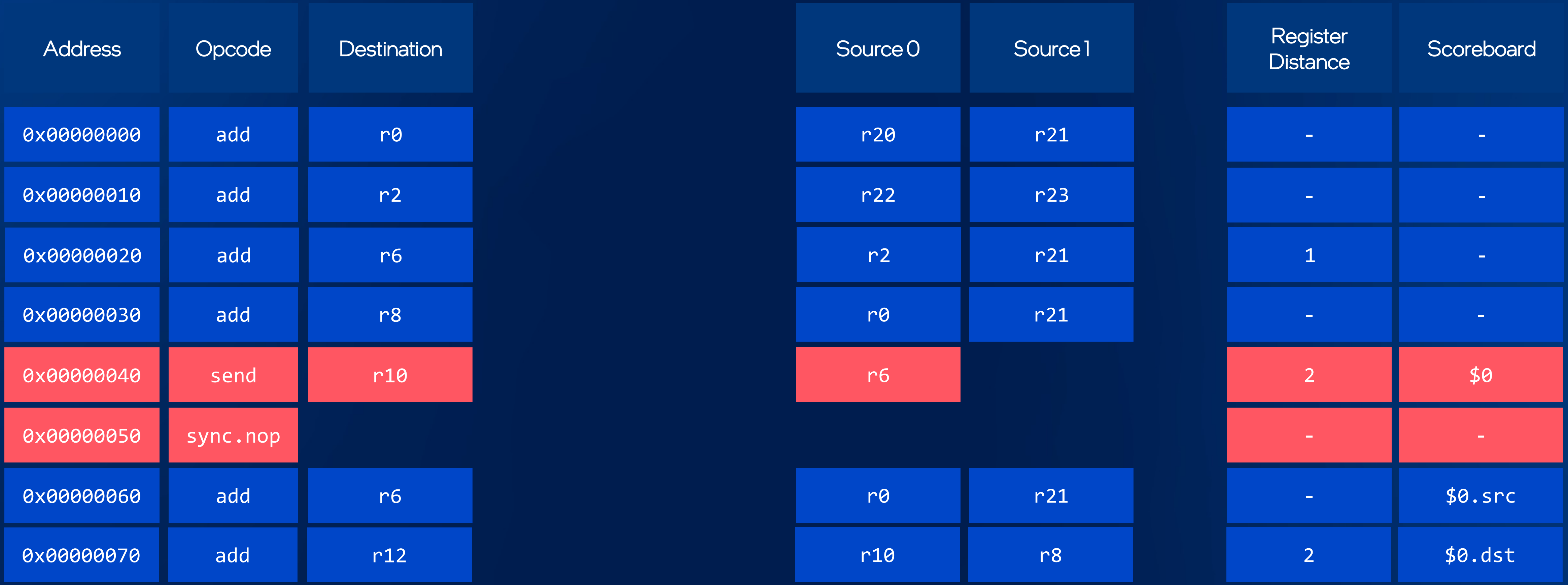


Instruction syntax

Scoreboard dependency

(simd) op d, s0, s1, s2 {rd, $sb}

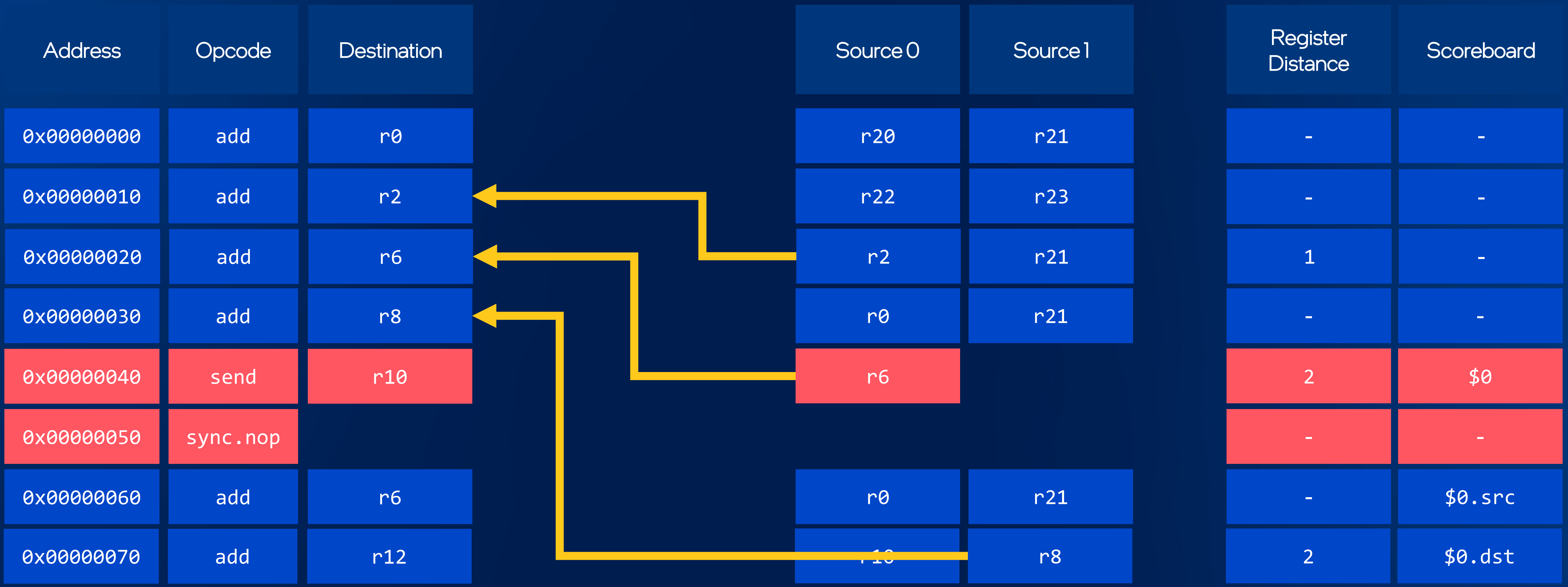
Register distance



Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

🟥 Out-of-order

🟦 In order

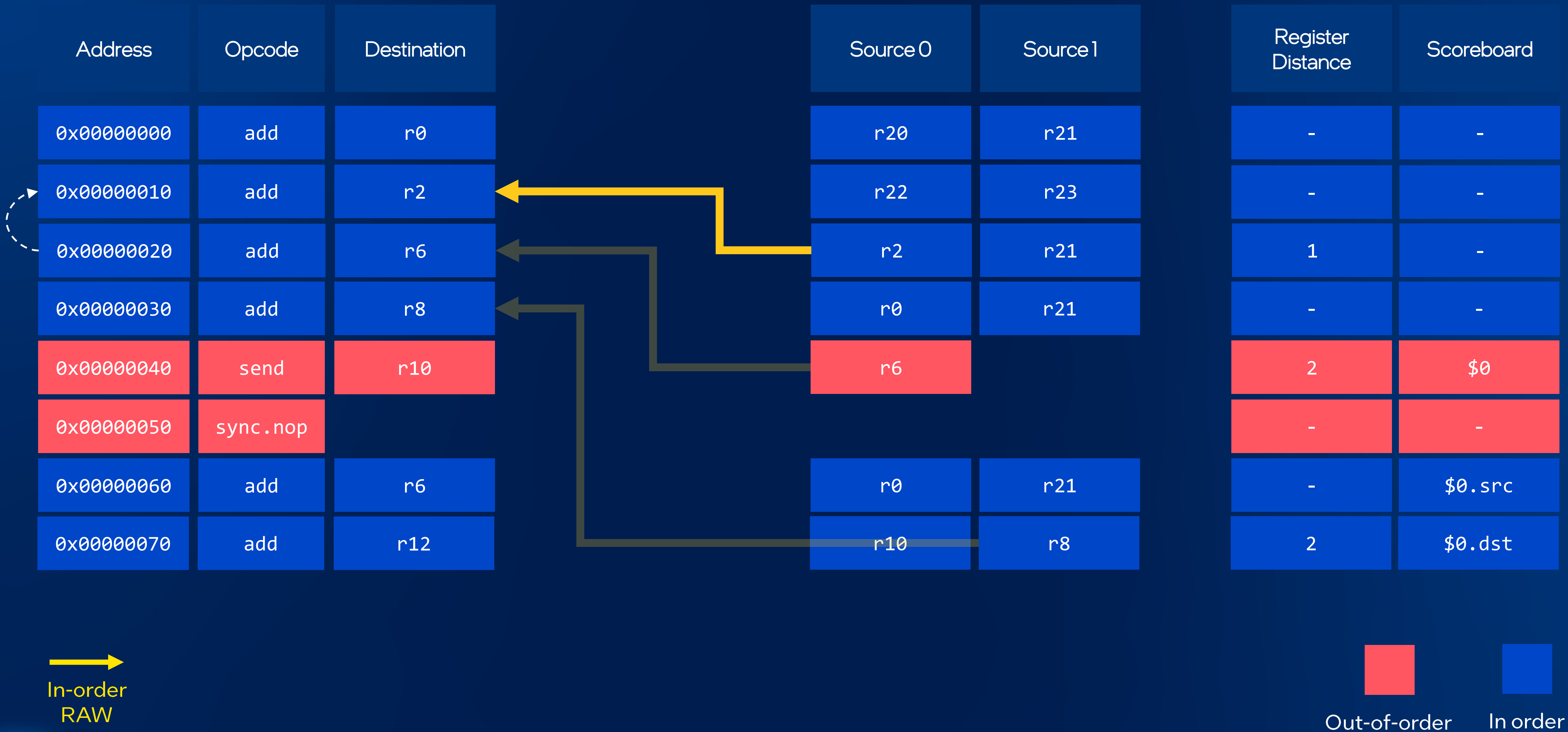


Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

→ In-order RAW

🟥 Out-of-order

🟦 In order

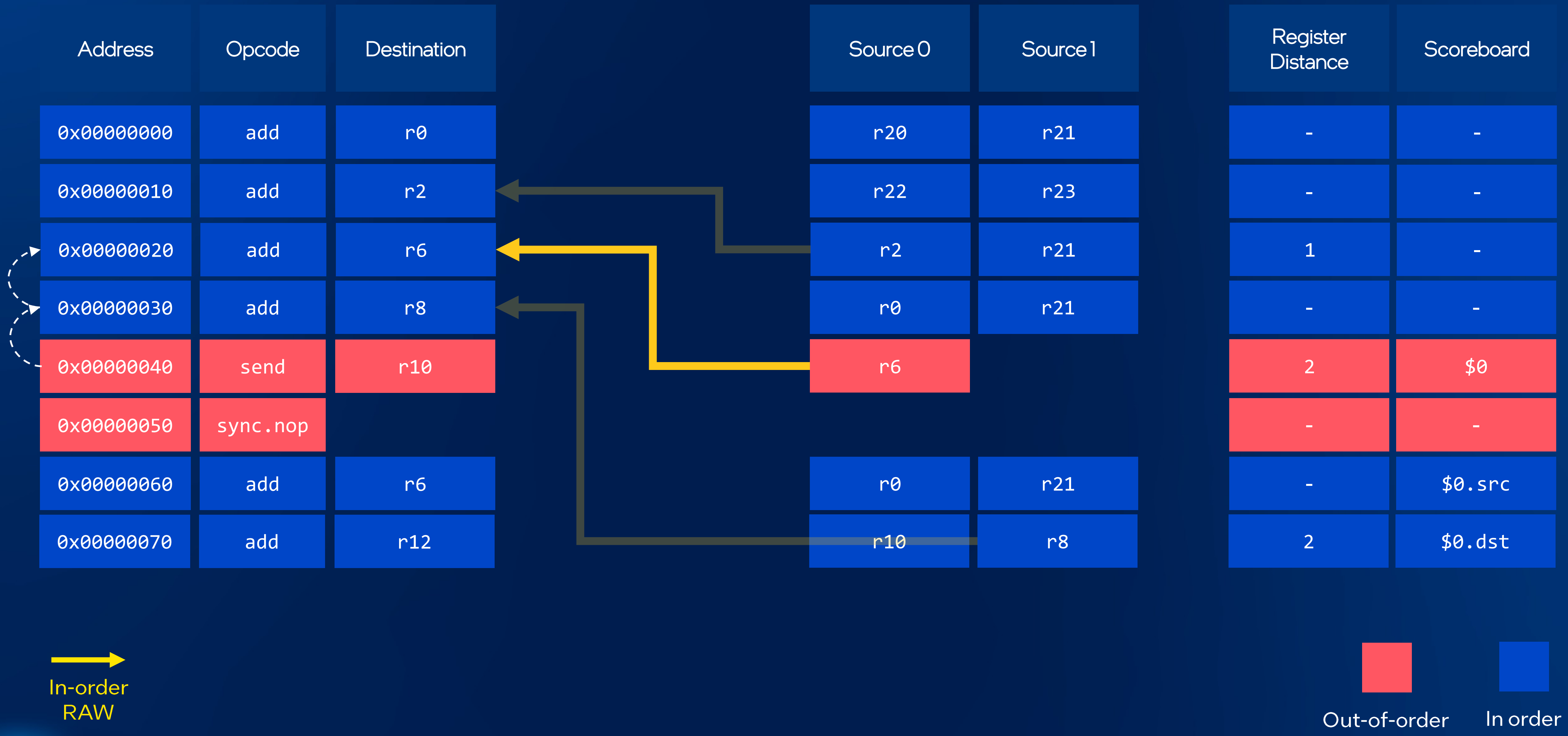


Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

In-order RAW

🟥 Out-of-order

🟦 In order

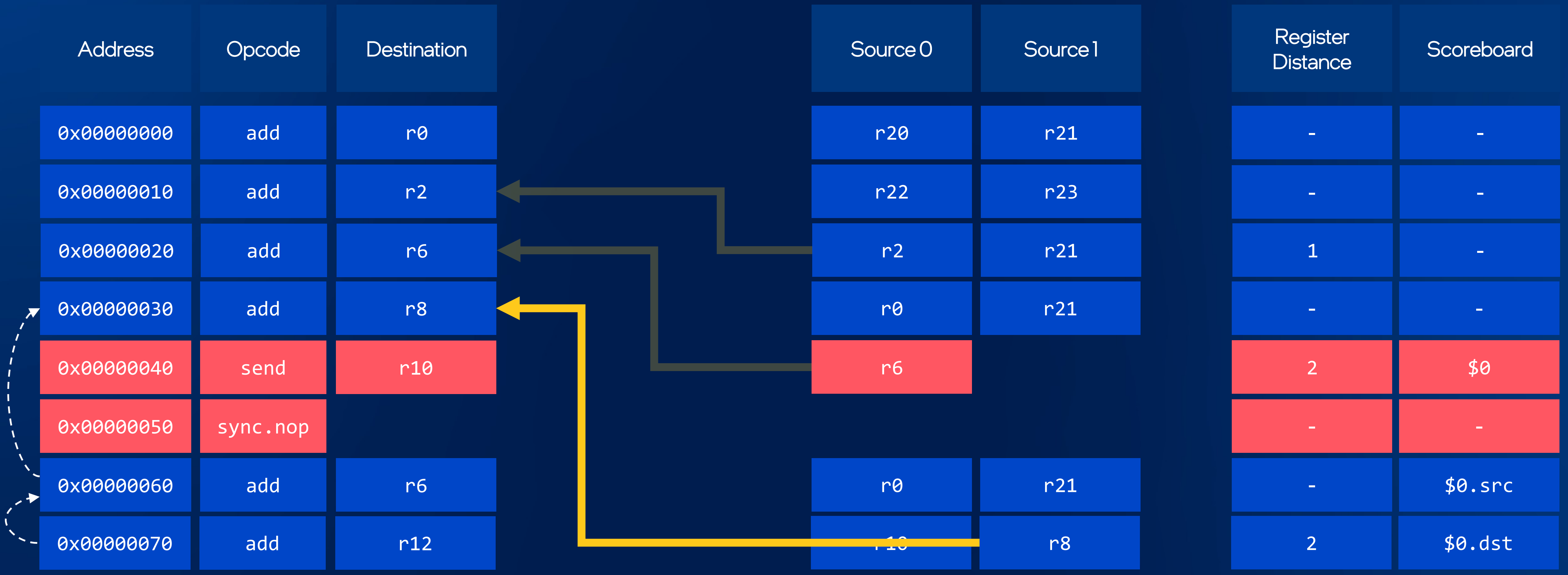


Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

In-order RAW

🟥 Out-of-order

🟦 In order

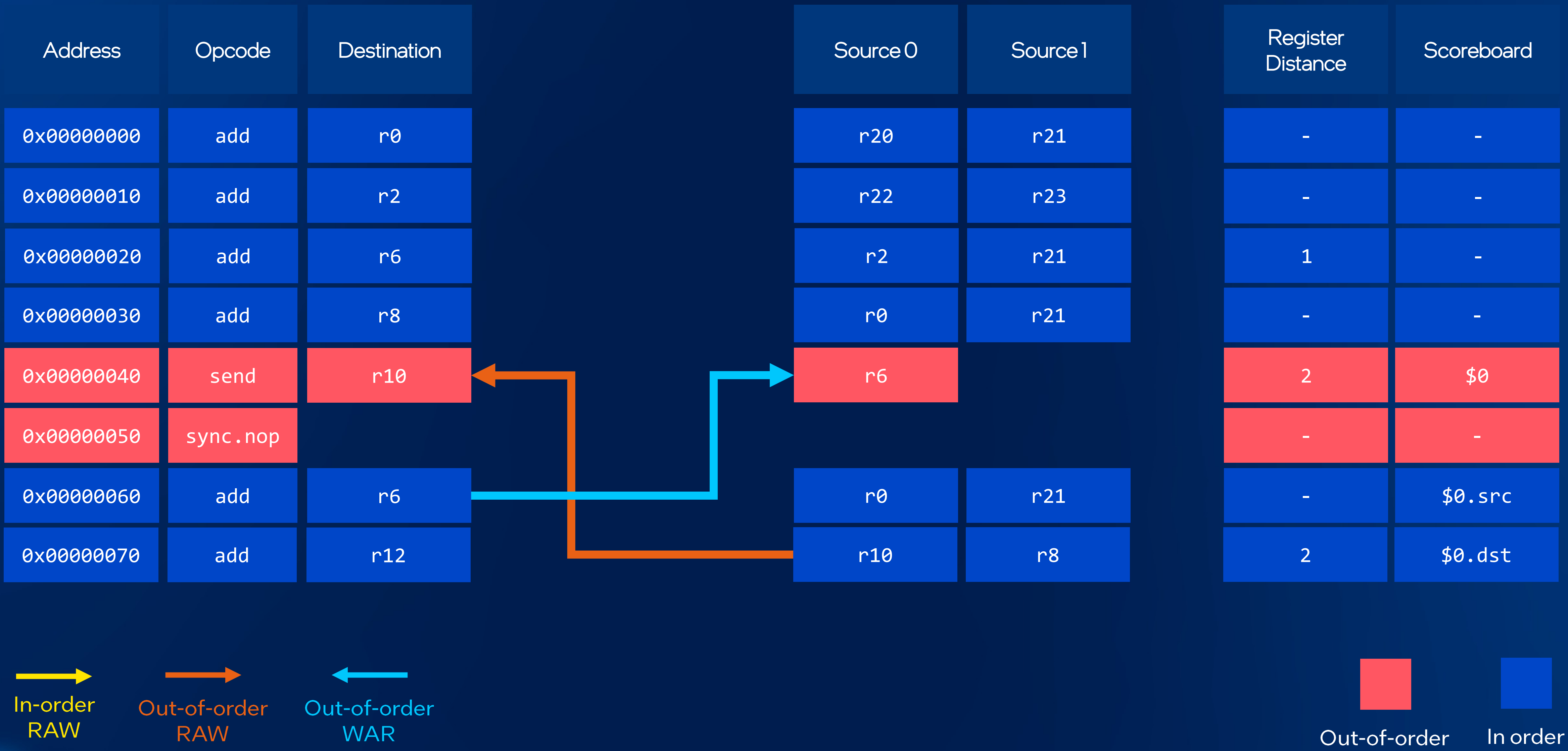


Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
`0x00000000`	`add`	`r0`	`r20`	`r21`	-	-
`0x00000010`	`add`	`r2`	`r22`	`r23`	-	-
`0x00000020`	`add`	`r6`	`r2`	`r21`	1	-
`0x00000030`	`add`	`r8`	`r0`	`r21`	-	-
`0x00000040`	`send`	`r10`	`r6`		2	`$0`
`0x00000050`	`sync.nop`				-	-
`0x00000060`	`add`	`r6`	`r0`	`r21`	-	`$0.src`
`0x00000070`	`add`	`r12`	`r10`	`r8`	2	`$0.dst`

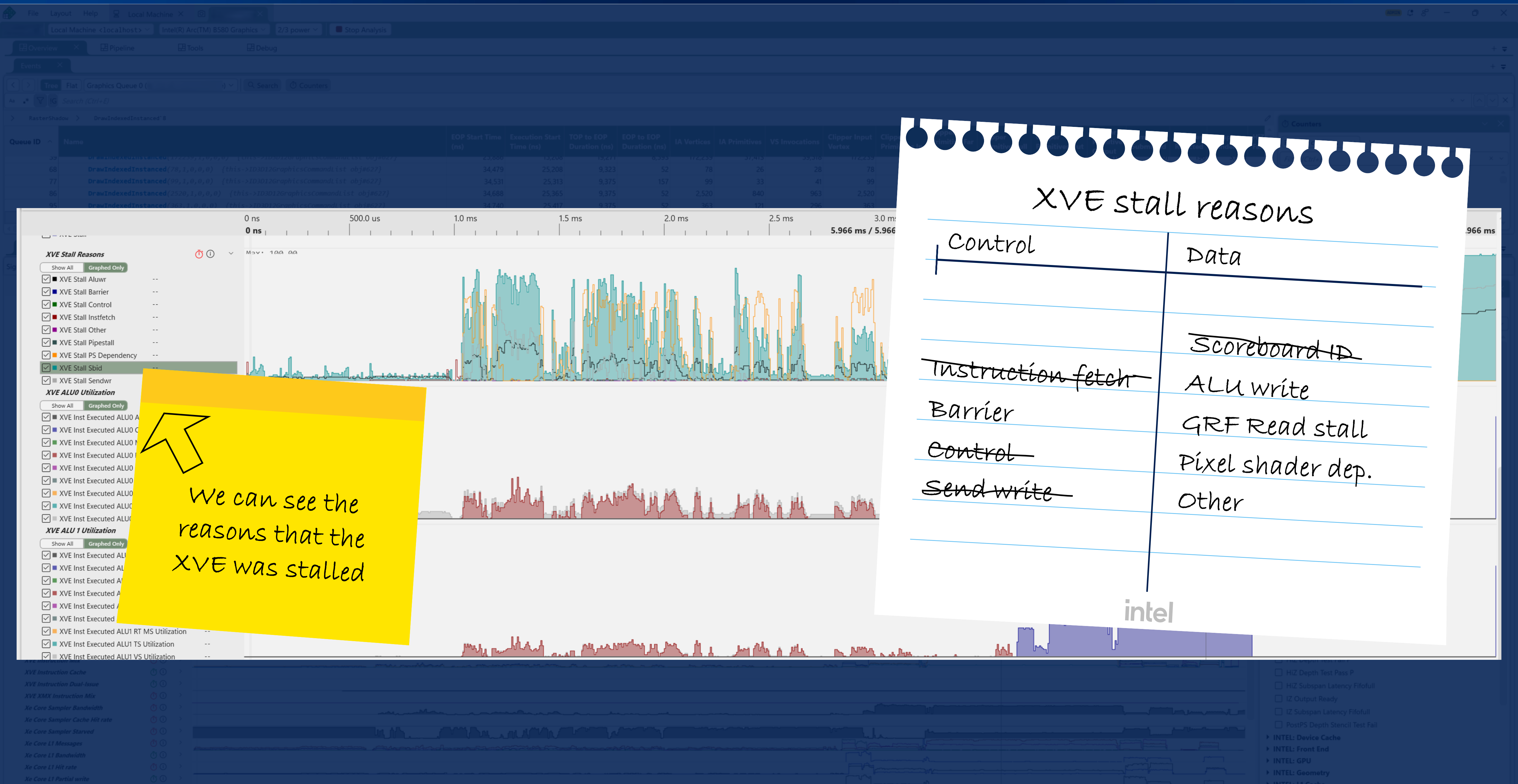
→ In-order RAW

🟥 Out-of-order

🟦 In order



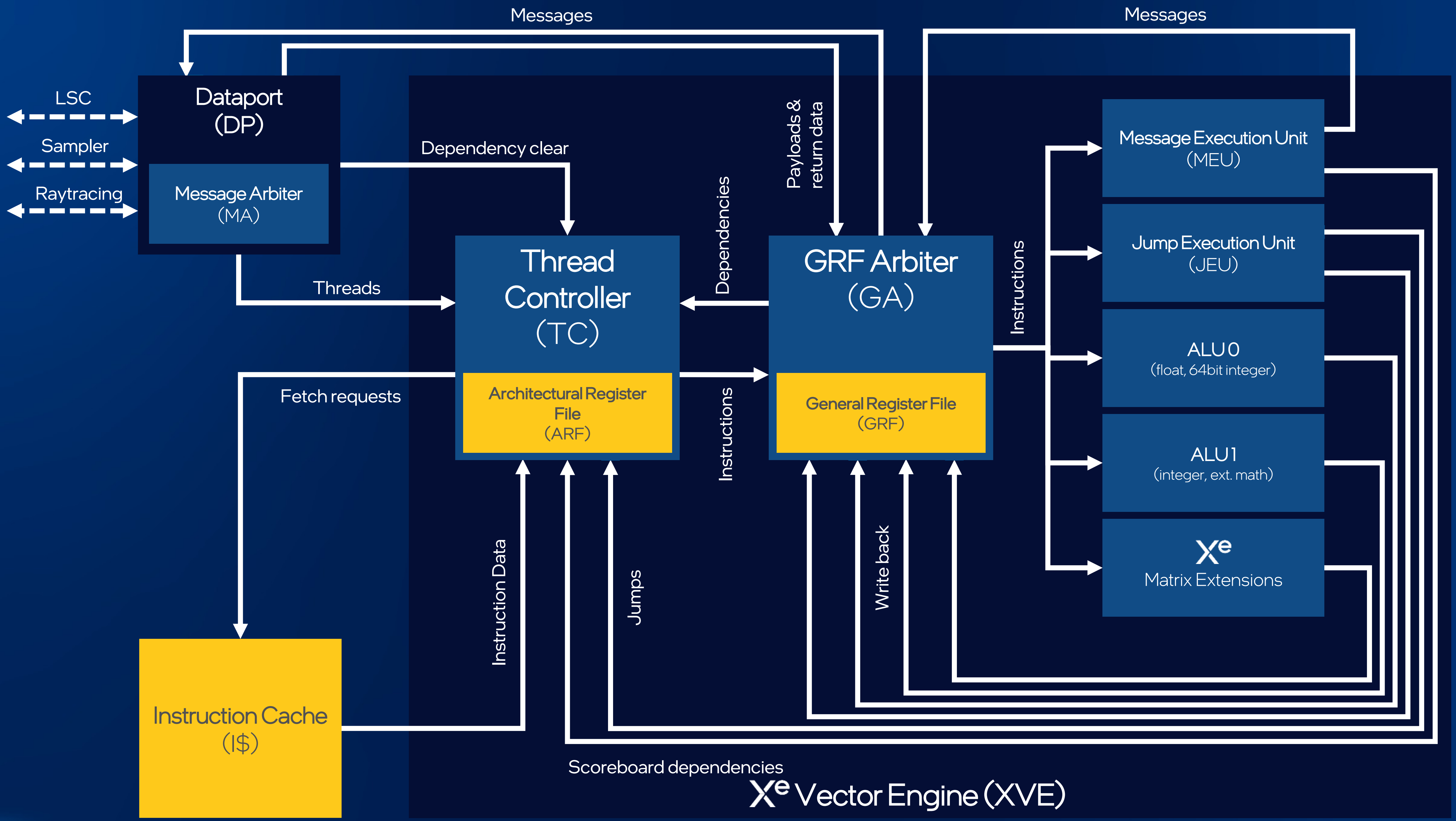
Address	Opcode	Destination	Source 0	Source 1	Register Distance	Scoreboard
0x00000000	add	r0	r20	r21	-	-
0x00000010	add	r2	r22	r23	-	-
0x00000020	add	r6	r2	r21	1	-
0x00000030	add	r8	r0	r21	-	-
0x00000040	send	r10	r6		2	$0
0x00000050	sync.nop				-	-
0x00000060	add	r6	r0	r21	-	$0.src
0x00000070	add	r12	r10	r8	2	$0.dst

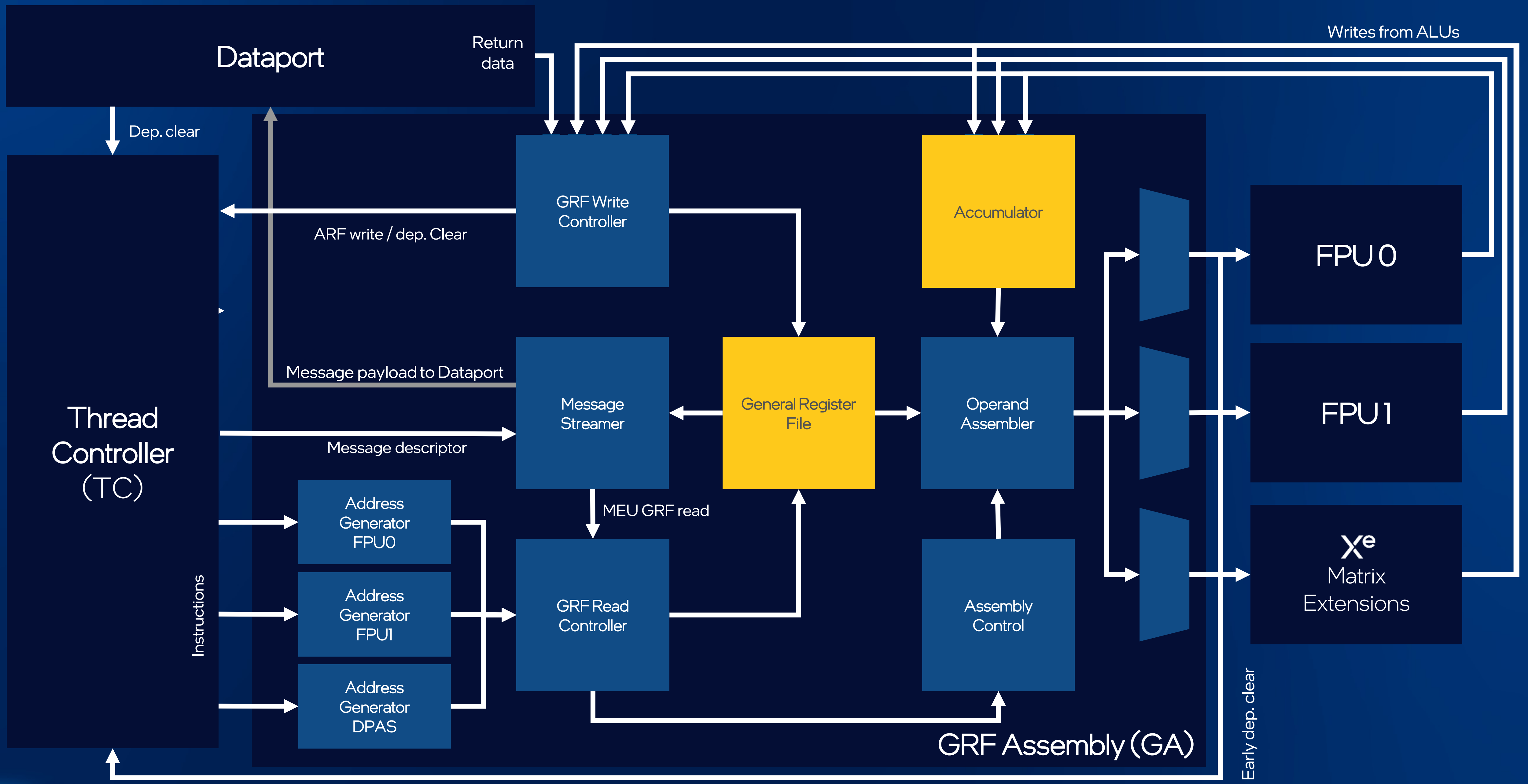


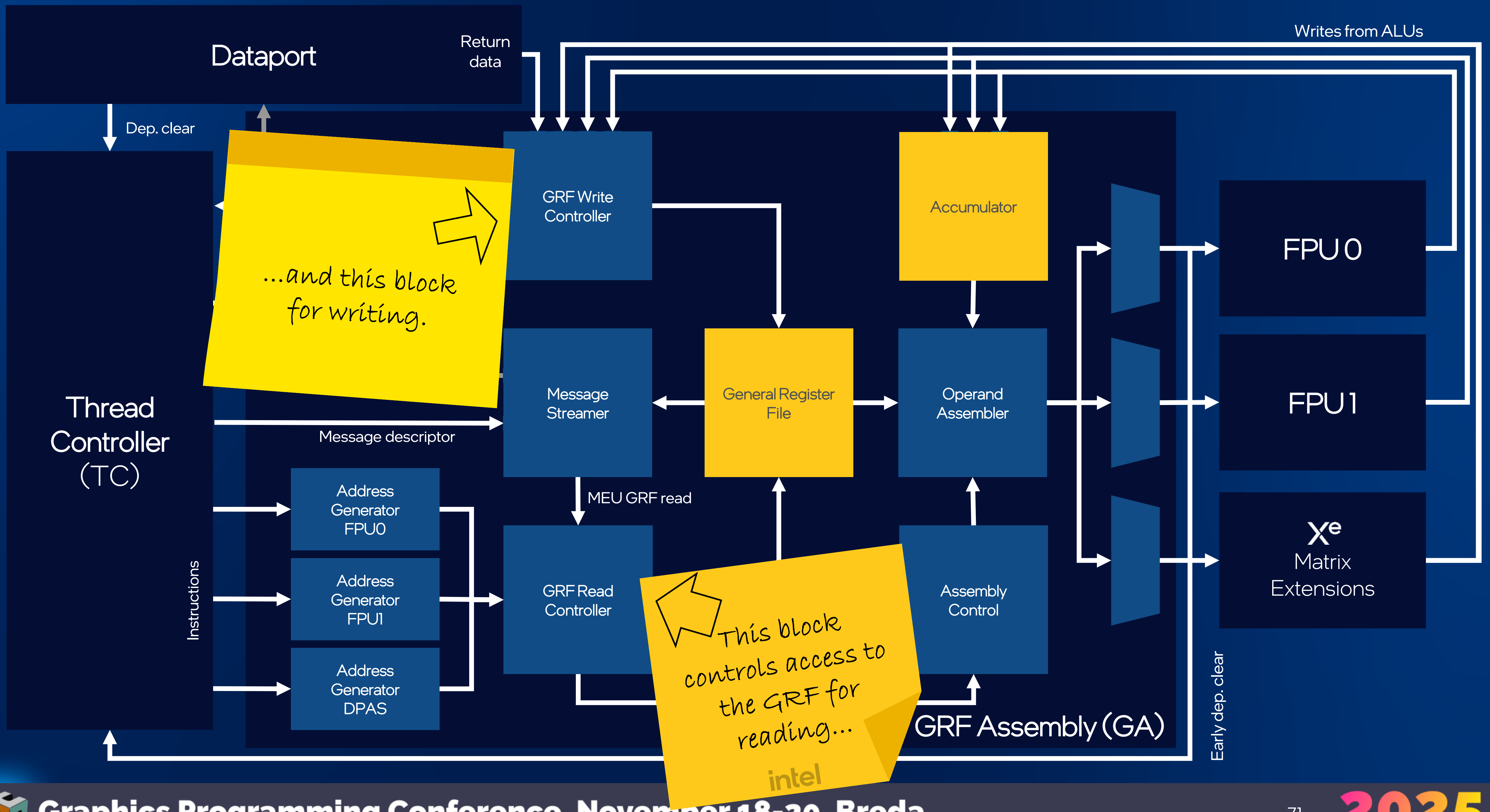
We can see the reasons that the XVE was stalled

XVE stall reasons

Control	Data
	~~Scoreboard ID~~
~~Instruction fetch~~	ALU write
Barrier	GRF Read stall
~~Control~~	Pixel shader dep.
~~Send write~~	Other

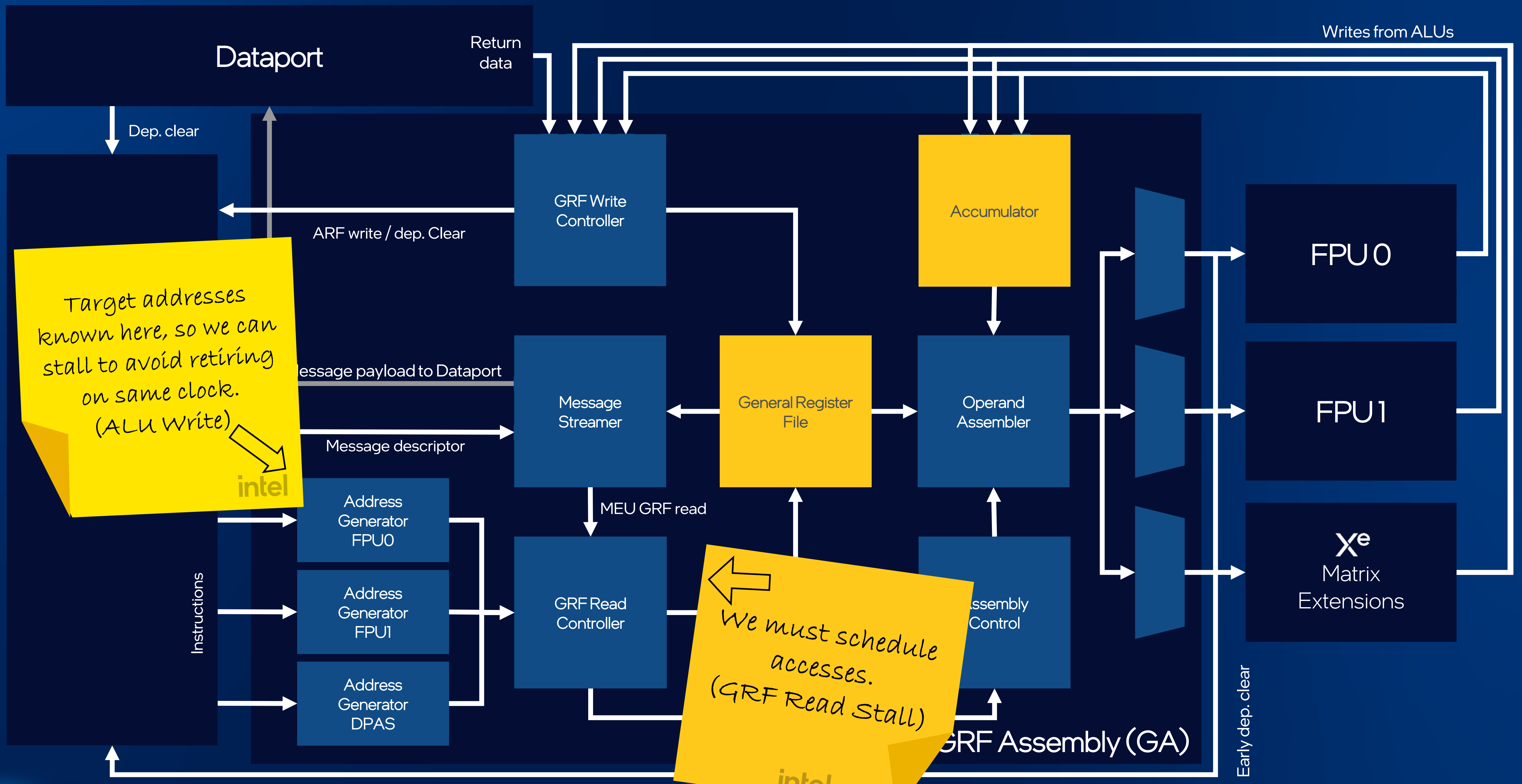

Messages
Messages
LSC
Sampler
Raytracing
Dataport
(DP)
Message Arbiter
(MA)
Dependency clear
Payloads & return data
Dependencies
Threads
Thread Controller
(TC)
GRF Arbiter
(GA)
Instructions
Message Execution Unit
(MEU)
Jump Execution Unit
(JEU)
ALU 0
(float, 64bit integer)
ALU 1
(integer, ext. math)
Xe
Matrix Extensions
Fetch requests
Architectural Register File
(ARF)
Instructions
General Register File
(GRF)
Instruction Data
Jumps
Write back
Instruction Cache
(I$)
Scoreboard dependencies
Xe Vector Engine (XVE)


Dataport
Return data
Writes from ALUs
Dep. clear
GRF Write Controller
Accumulator
ARF write / dep. Clear
FPU 0
Thread Controller (TC)
Message payload to Dataport
Message Streamer
General Register File
Operand Assembler
FPU 1
Message descriptor
Address Generator FPU0
MEU GRF read
Xe Matrix Extensions
Address Generator FPU1
GRF Read Controller
Assembly Control
Instructions
Address Generator DPAS
GRF Assembly (GA)
Early dep. clear



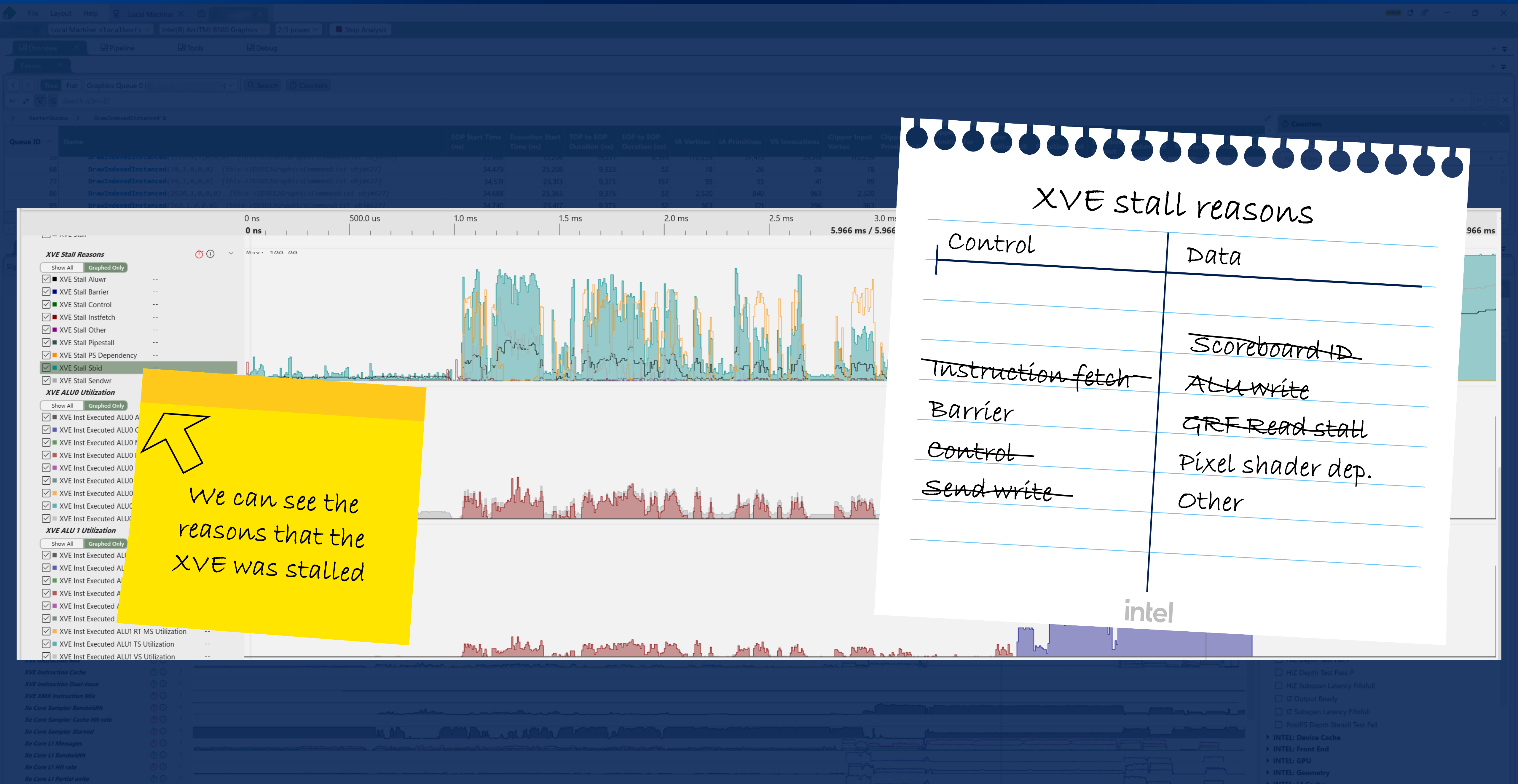
…and this block for writing.

This block controls access to the GRF for reading…



Target addresses known here, so we can stall to avoid retiring on same clock. (ALU Write)

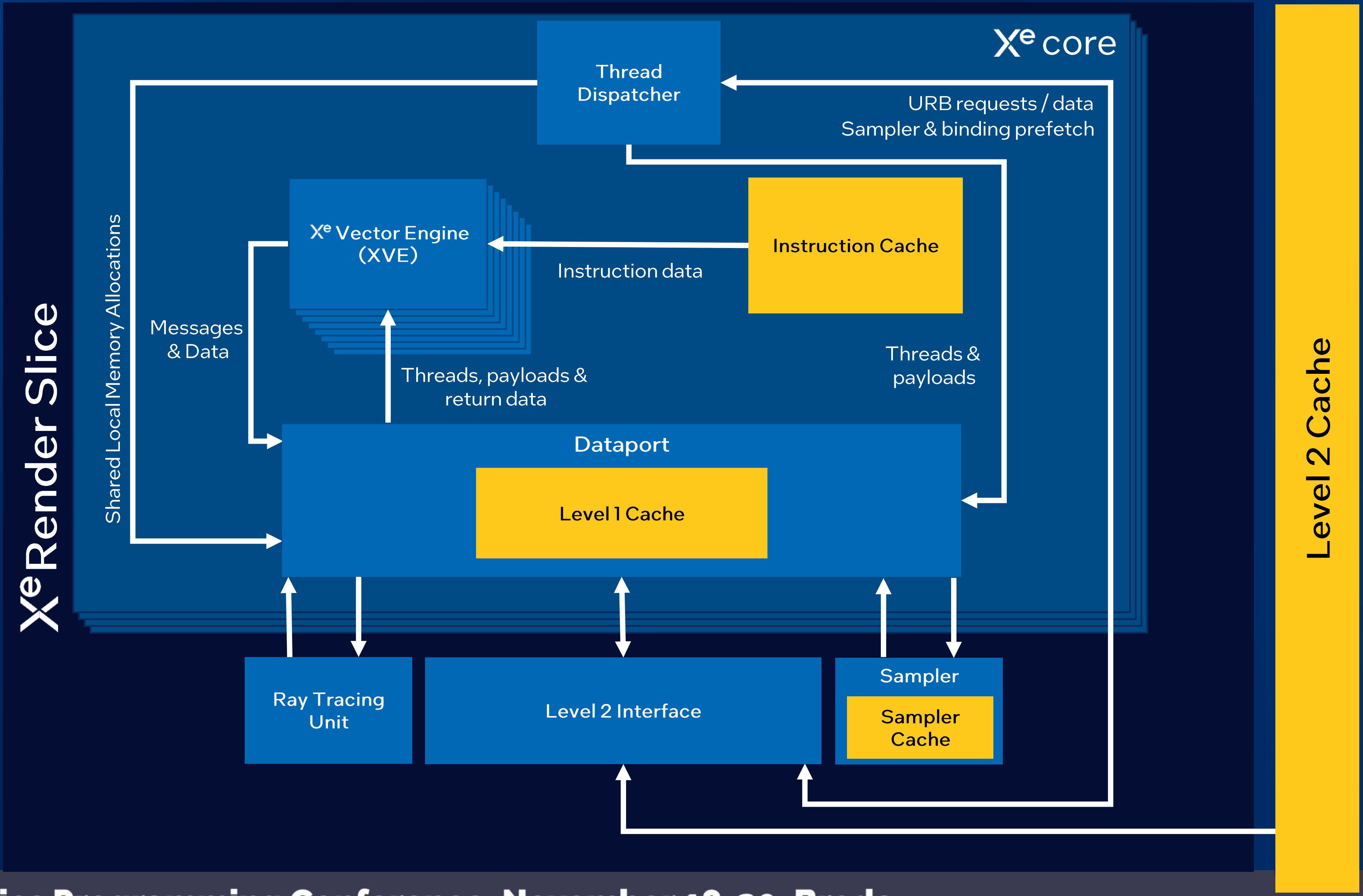
We must schedule accesses. (GRF Read Stall)

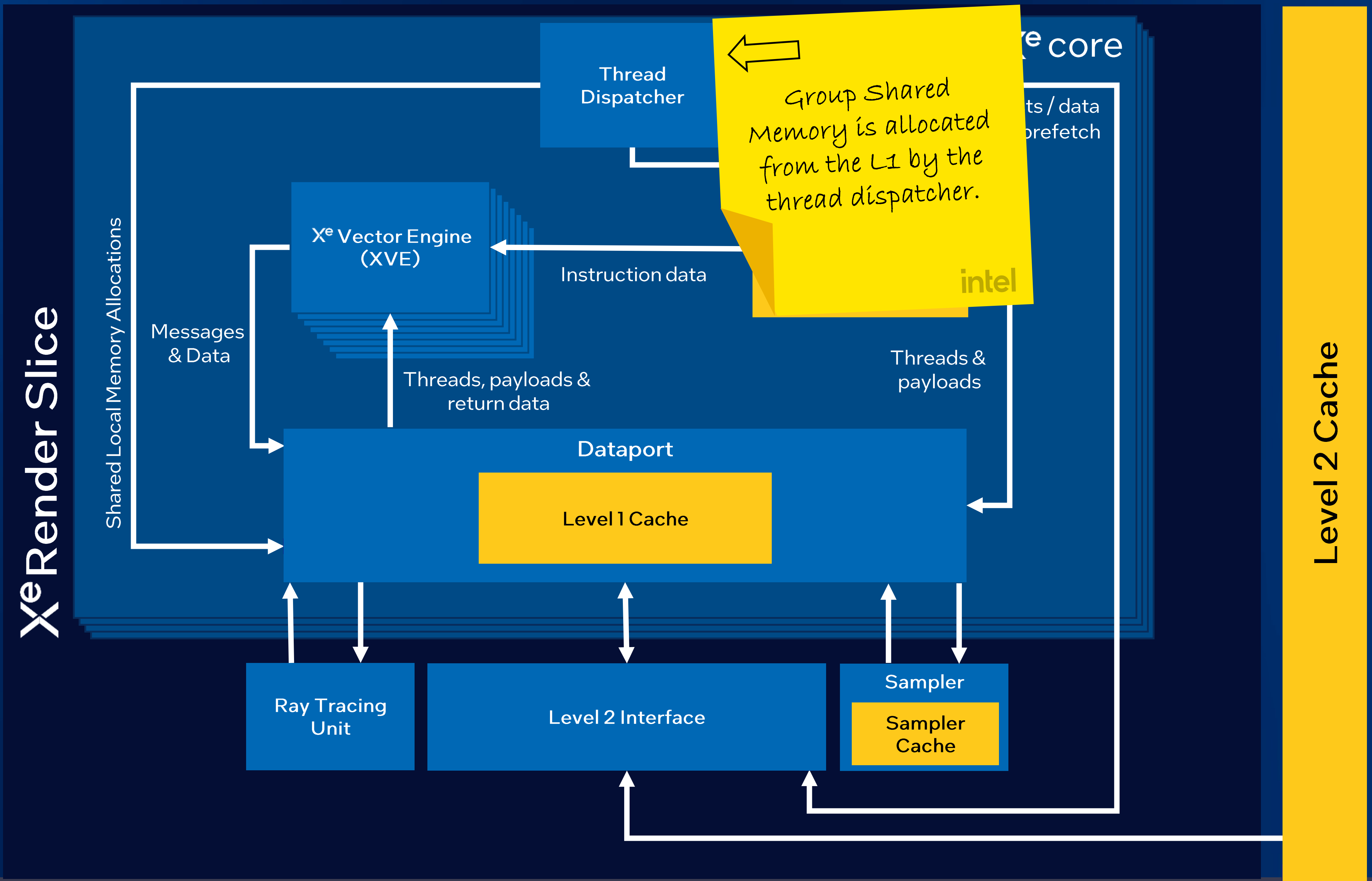


We can see the reasons that the XVE was stalled

XVE stall reasons

Control	Data
	~~Scoreboard ID~~
~~Instruction fetch~~	~~ALU write~~
Barrier	~~GRF Read stall~~
~~Control~~	Pixel shader dep.
~~Send write~~	Other


Xe core
Thread Dispatcher
URB requests / data
Sampler & binding prefetch
Xe Vector Engine (XVE)
Instruction data
Instruction Cache
Messages & Data
Threads, payloads & return data
Threads & payloads
Shared Local Memory Allocations
Xe Render Slice
Dataport
Level 1 Cache
Ray Tracing Unit
Level 2 Interface
Sampler
Sampler Cache
Level 2 Cache



X^e Render Slice

Shared Local Memory Allocations

Thread Dispatcher

Group Shared Memory is allocated from the L1 by the thread dispatcher.

e core

ts / data
prefetch

X^e Vector Engine (XVE)

Messages & Data

Instruction data

Threads, payloads & return data

Threads & payloads

Dataport

Level 1 Cache

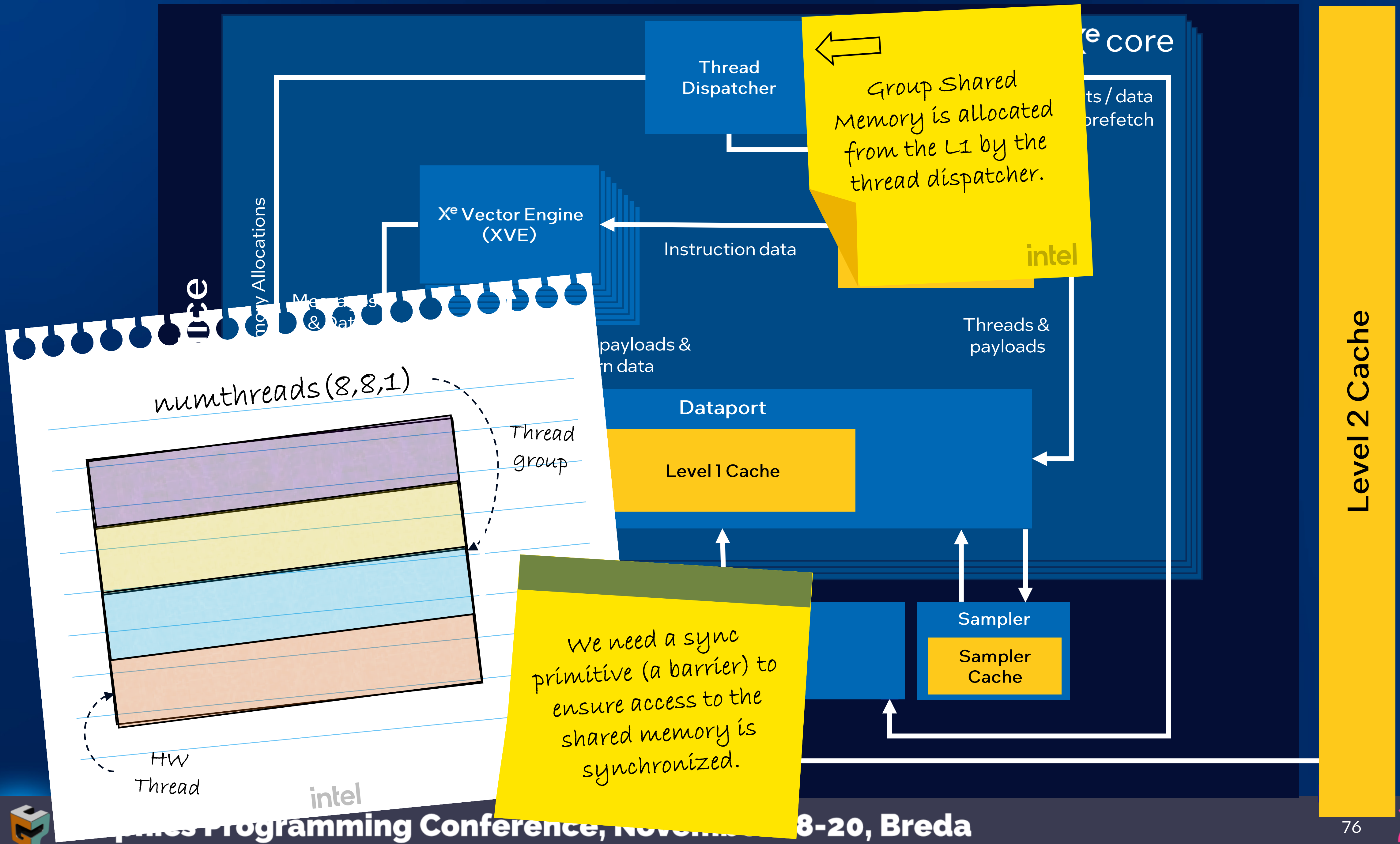
Ray Tracing Unit

Level 2 Interface

Sampler

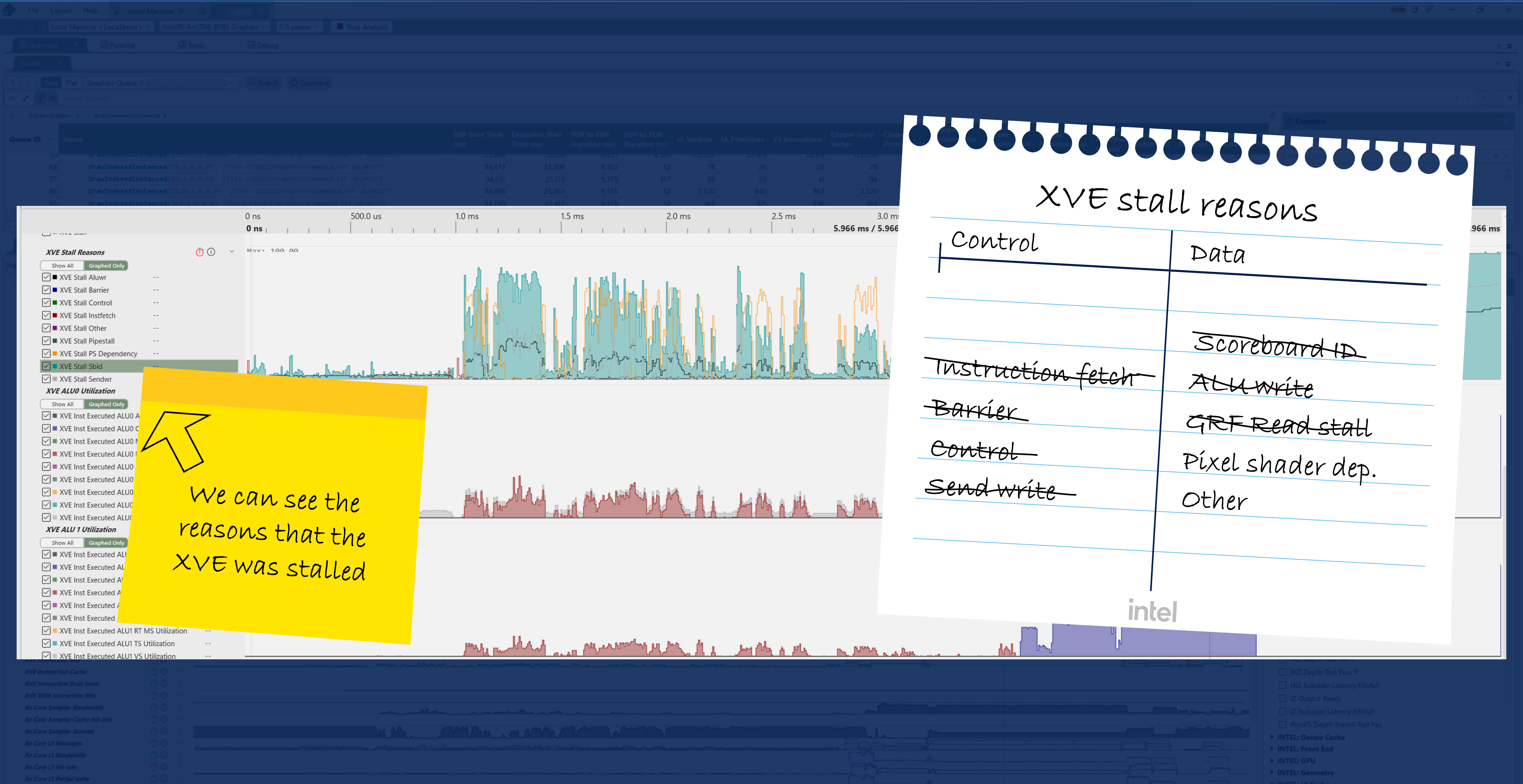
Sampler Cache

Level 2 Cache



Group Shared Memory is allocated from the L1 by the thread dispatcher.

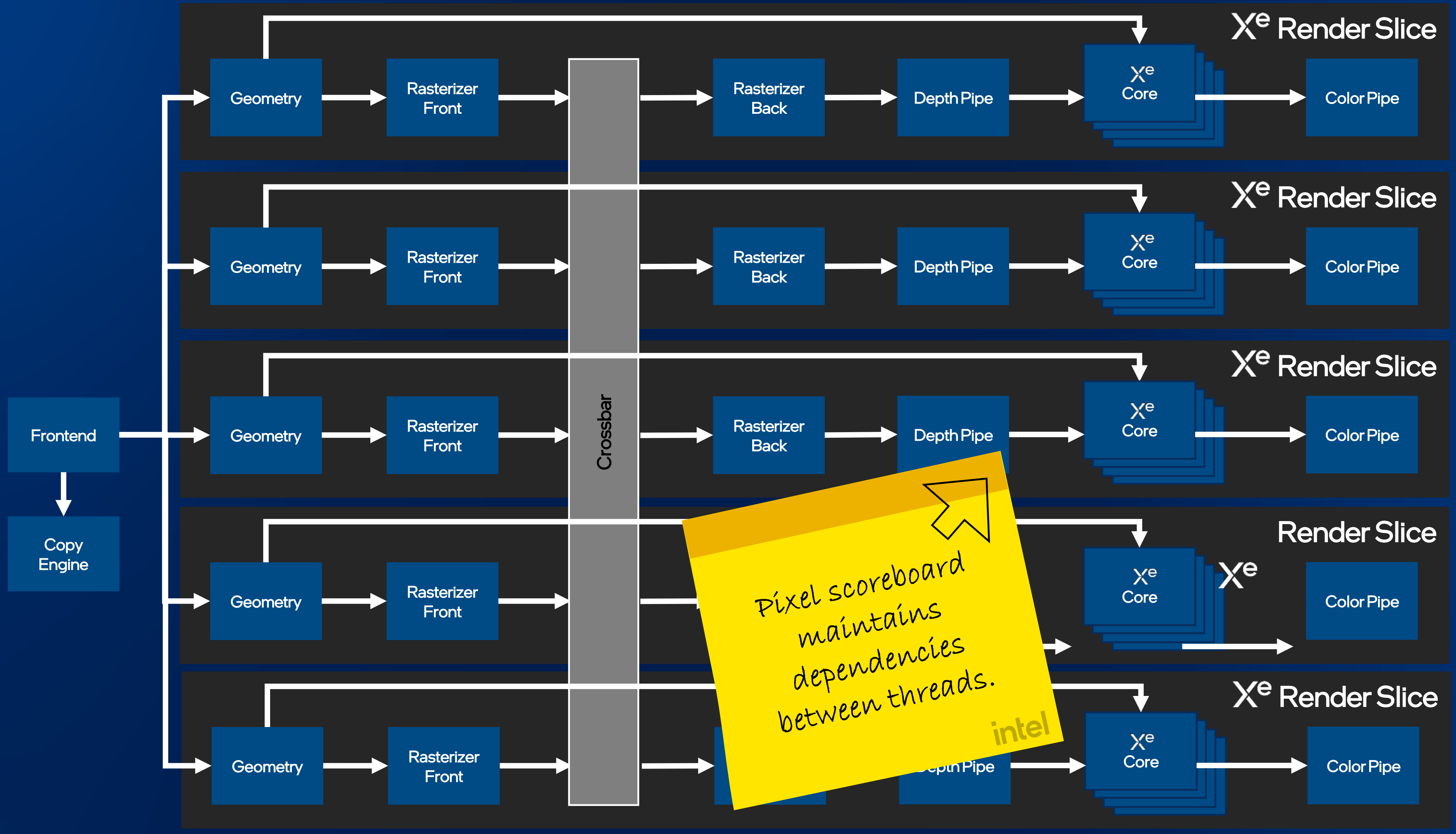
We need a sync primitive (a barrier) to ensure access to the shared memory is synchronized.



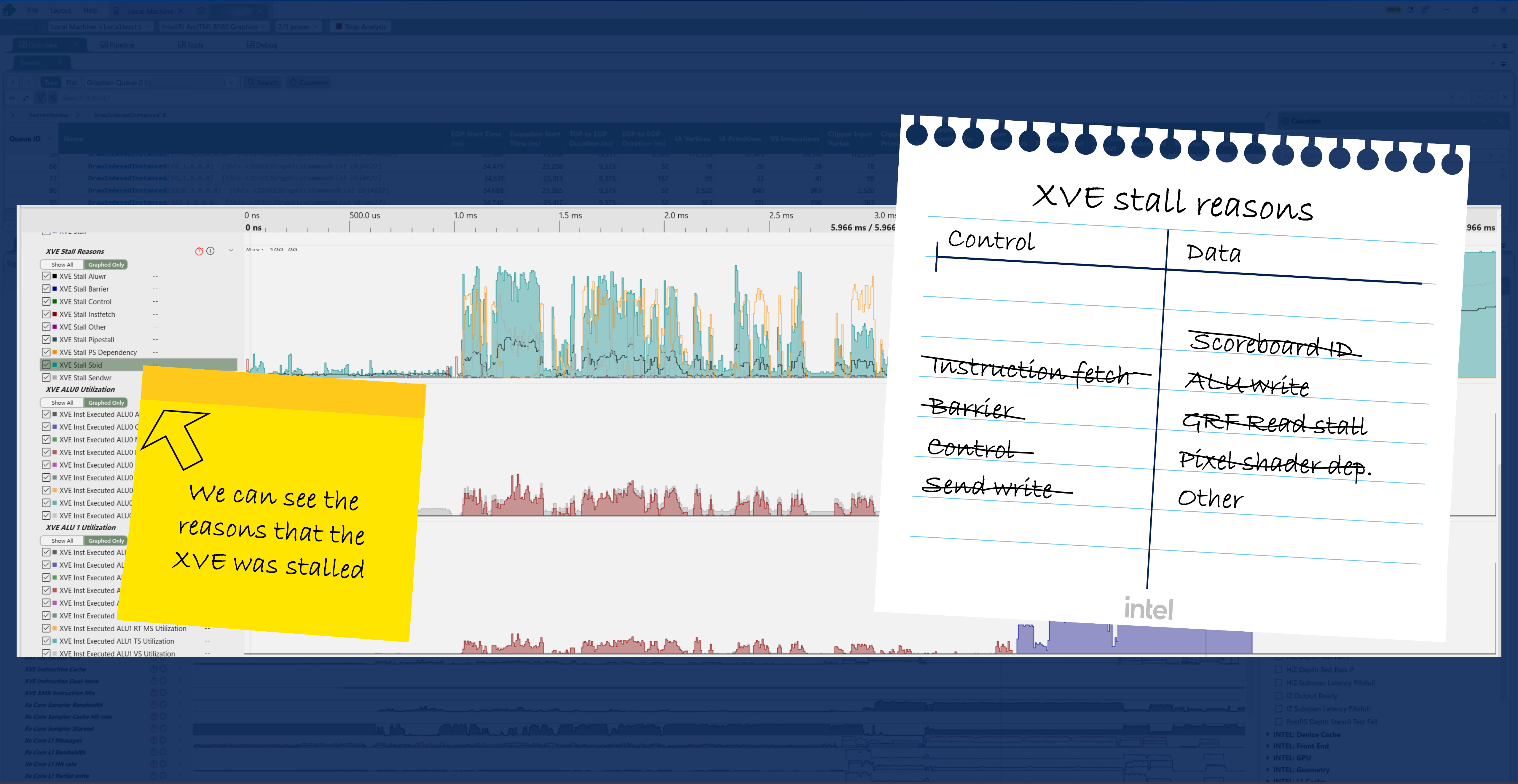
We can see the reasons that the XVE was stalled

XVE stall reasons

Control	Data
	~~Scoreboard ID~~
~~Instruction fetch~~	~~ALU write~~
~~Barrier~~	~~GRF Read stall~~
~~Control~~	Pixel shader dep.
~~Send write~~	Other



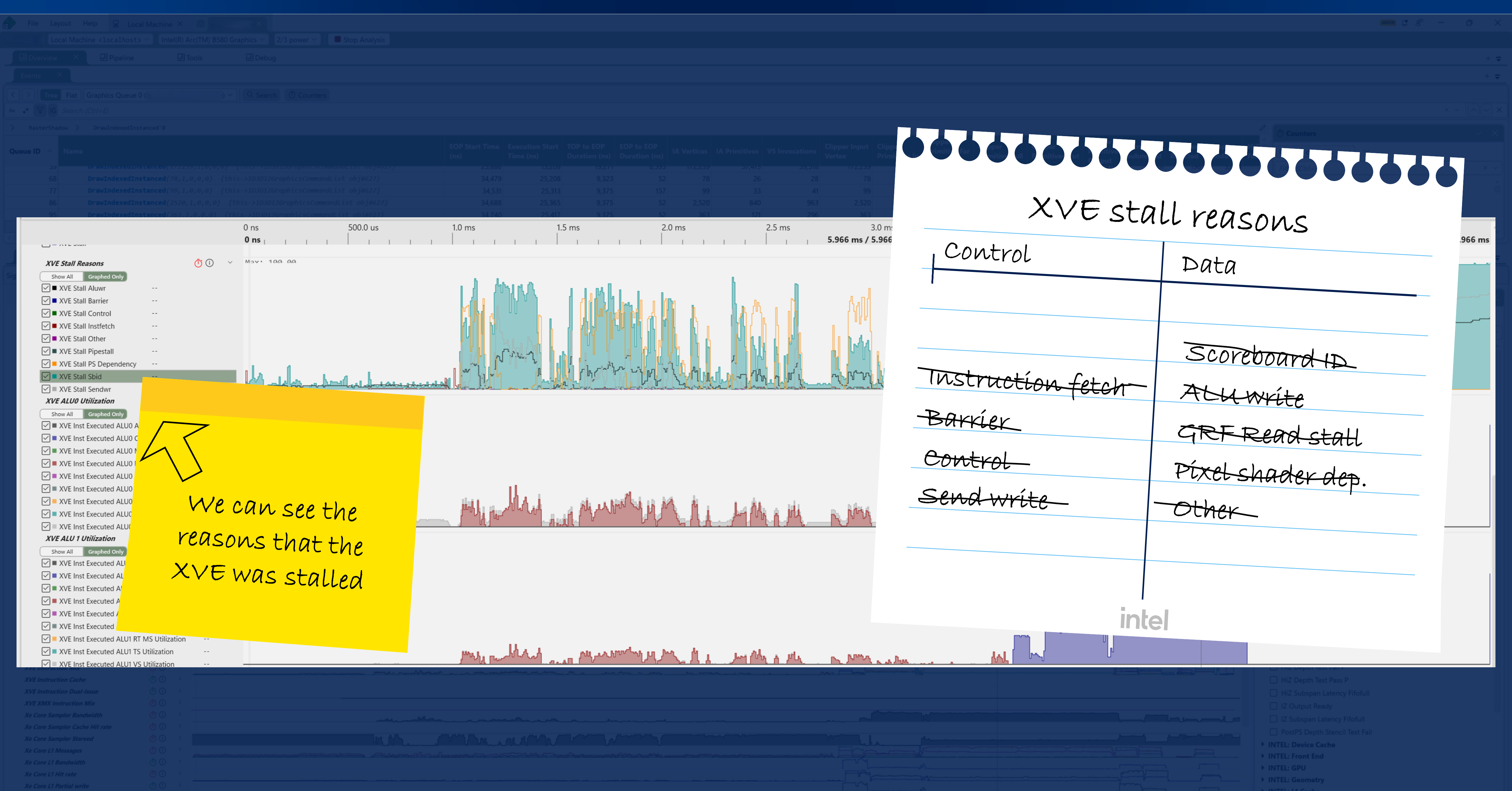
Pixel scoreboard maintains dependencies between threads.



We can see the reasons that the XVE was stalled

XVE stall reasons

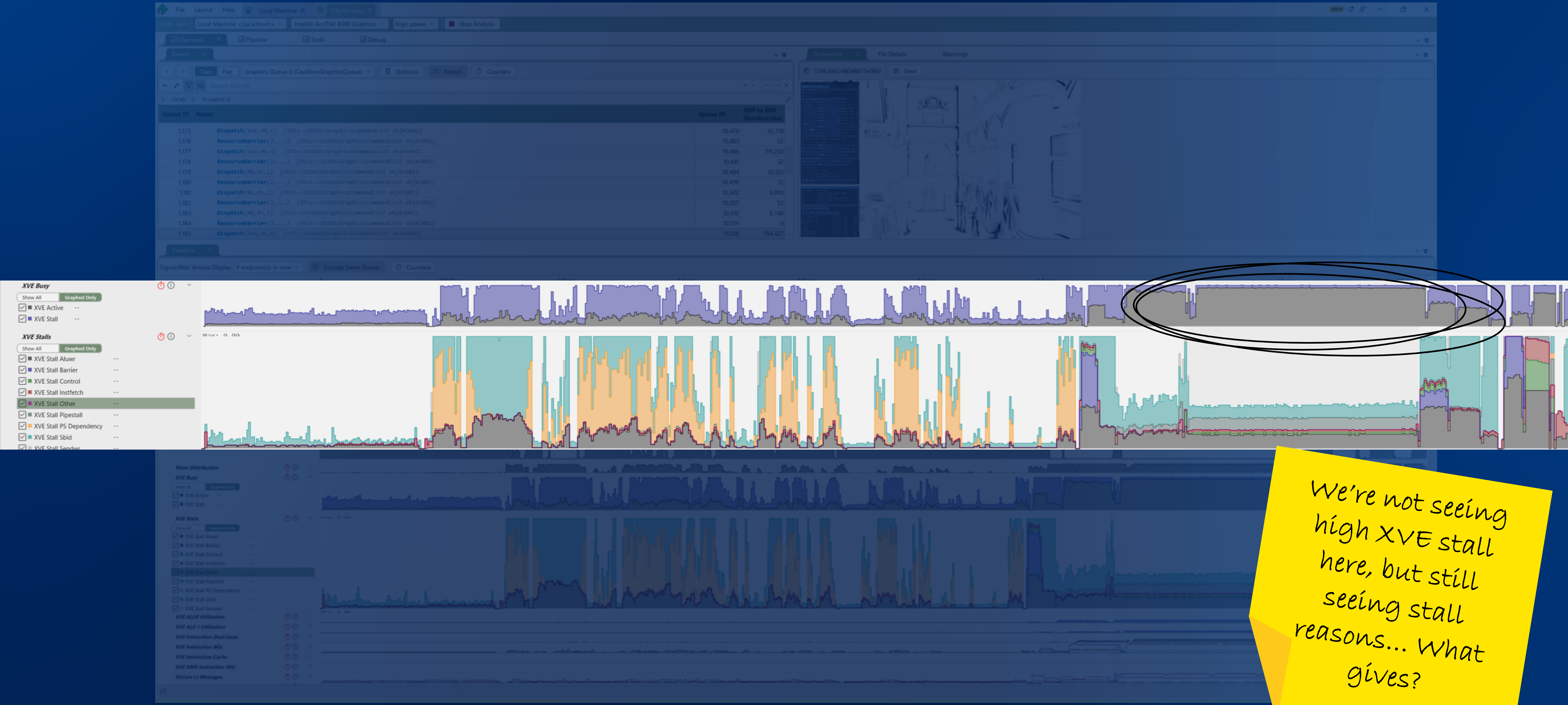
Control	Data
	~~Scoreboard ID~~
~~Instruction fetch~~	~~ALU write~~
~~Barrier~~	~~GRF Read stall~~
~~Control~~	~~Pixel shader dep.~~
~~Send write~~	Other



We can see the reasons that the XVE was stalled

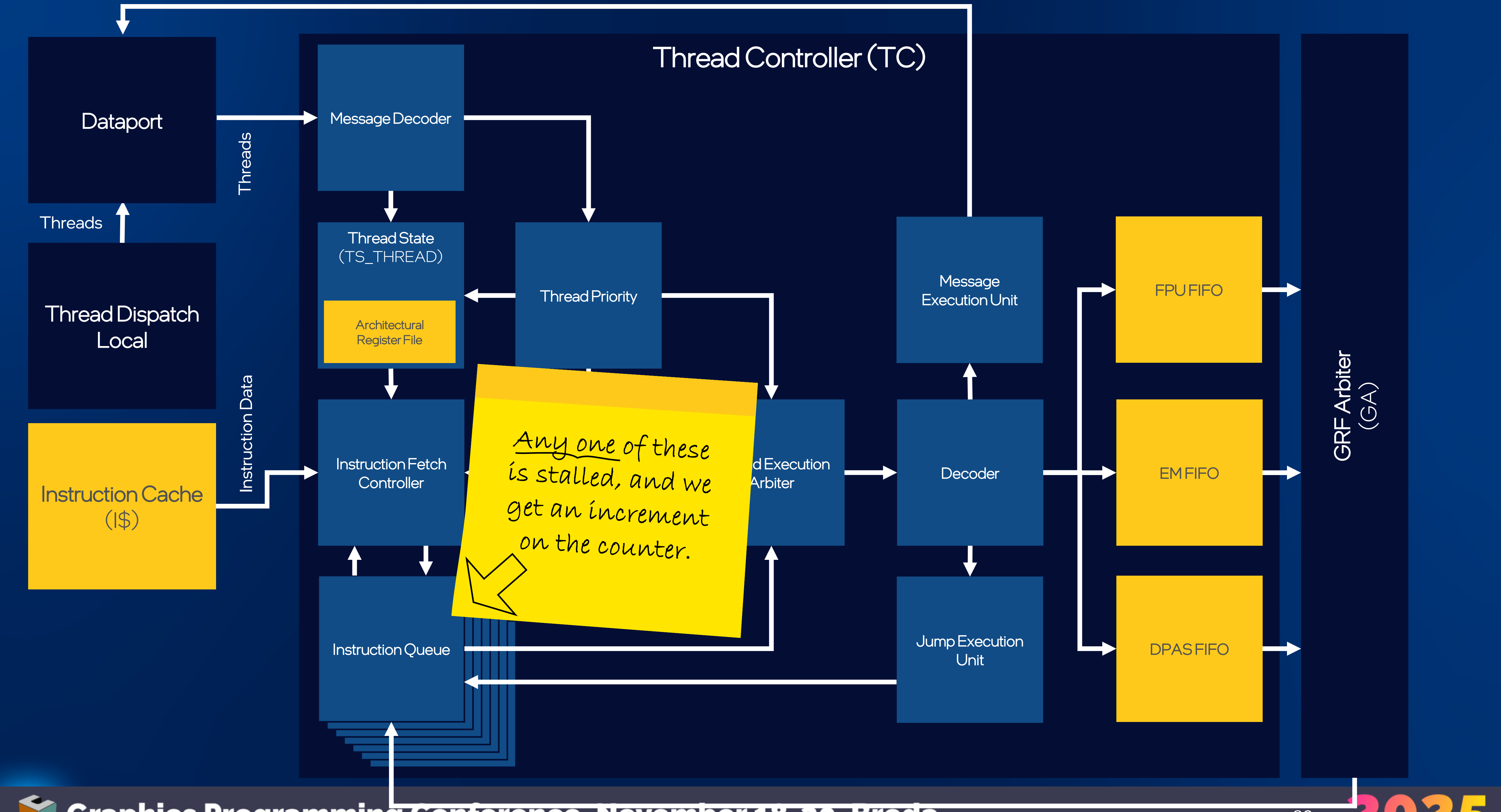
XVE stall reasons

Control	Data
	~~Scoreboard ID~~
~~Instruction fetch~~	~~ALU write~~
~~Barrier~~	~~GRF Read stall~~
~~Control~~	~~Pixel shader dep.~~
~~Send write~~	~~Other~~



We're not seeing high XVE stall here, but still seeing stall reasons... What gives?

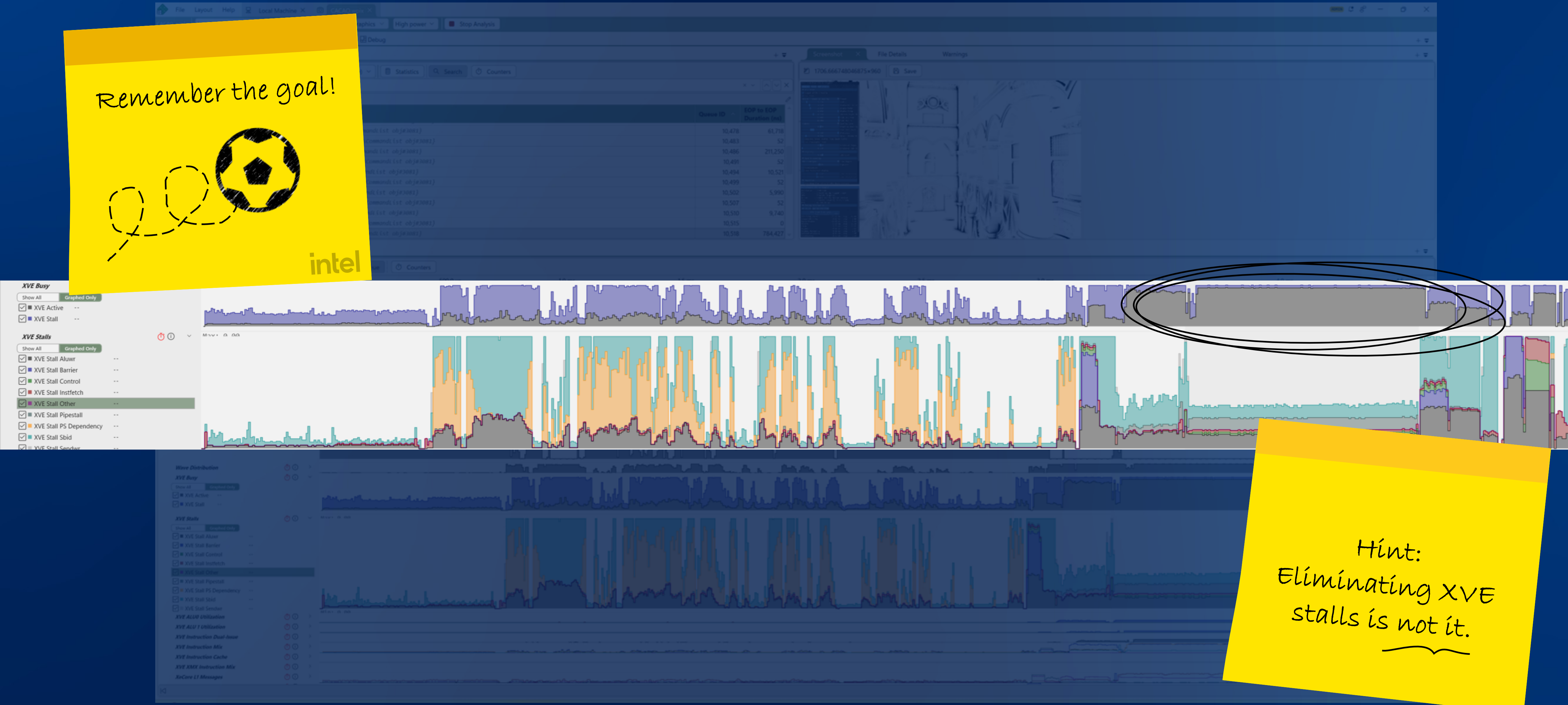
intel



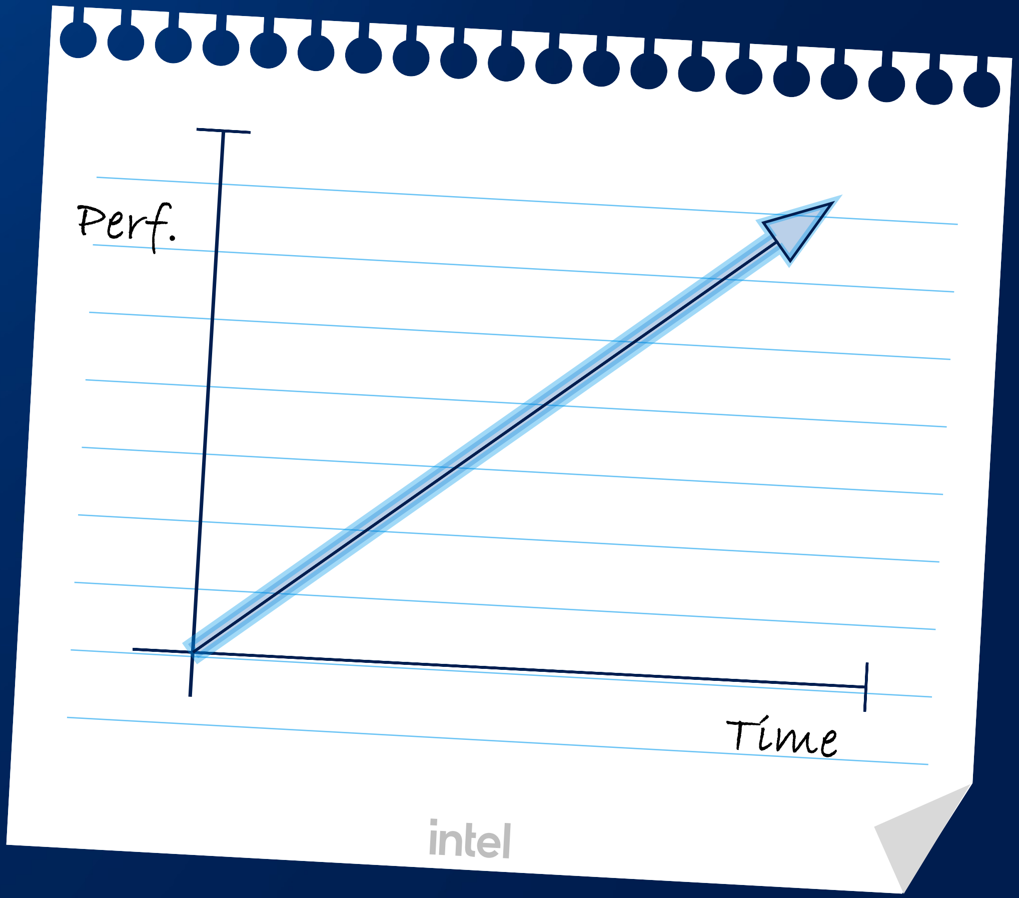
Any one of these is stalled, and we get an increment on the counter.

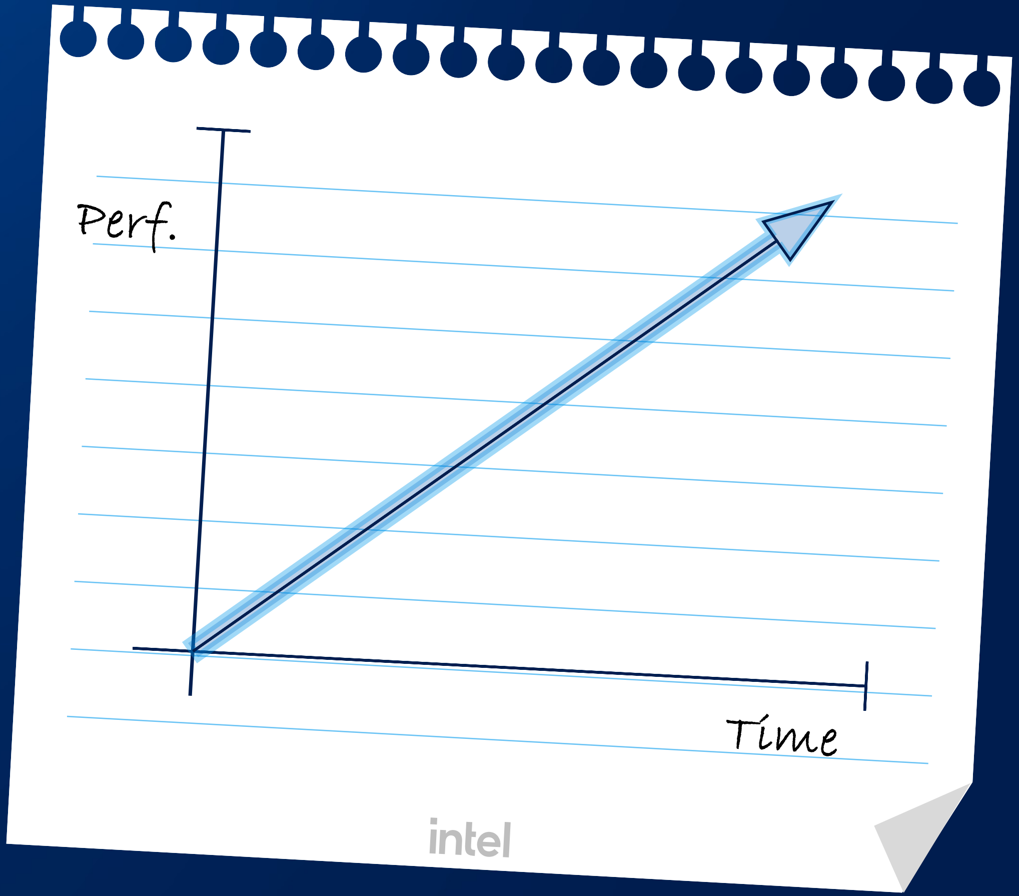
Remember the goal!

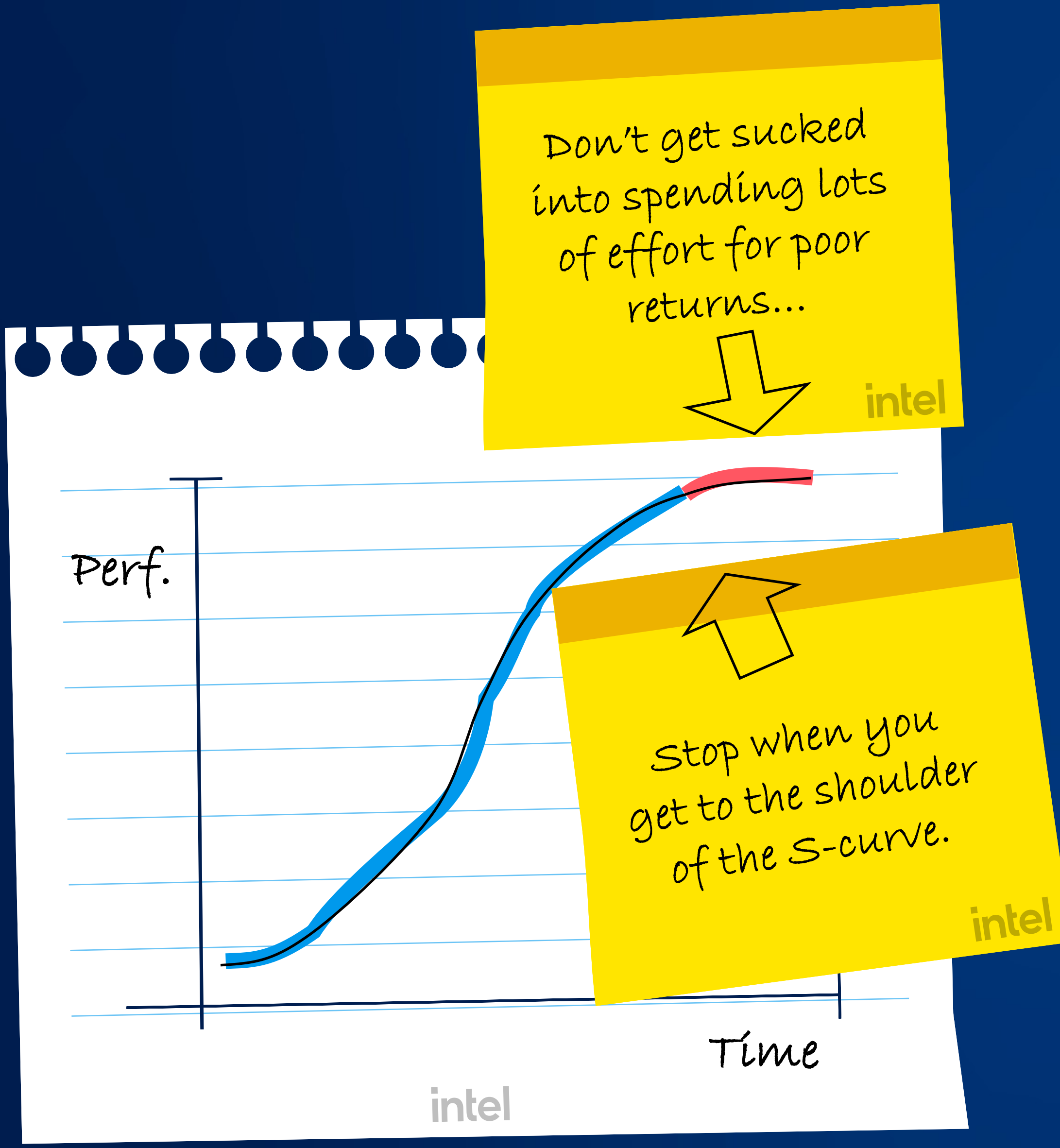
intel



Hint:
Eliminating XVE
stalls is not it.


Perf.
Time
intel





Don't get sucked into spending lots of effort for poor returns...

Stop when you get to the shoulder of the S-curve.

To do list:

- Know the optimization process
- Define goals, and focus on them
- Know my tools
- Understand the substrate
- Aim for 100%, but stop at ~95%
- Optimize regularly

askwinpix@microsoft.com

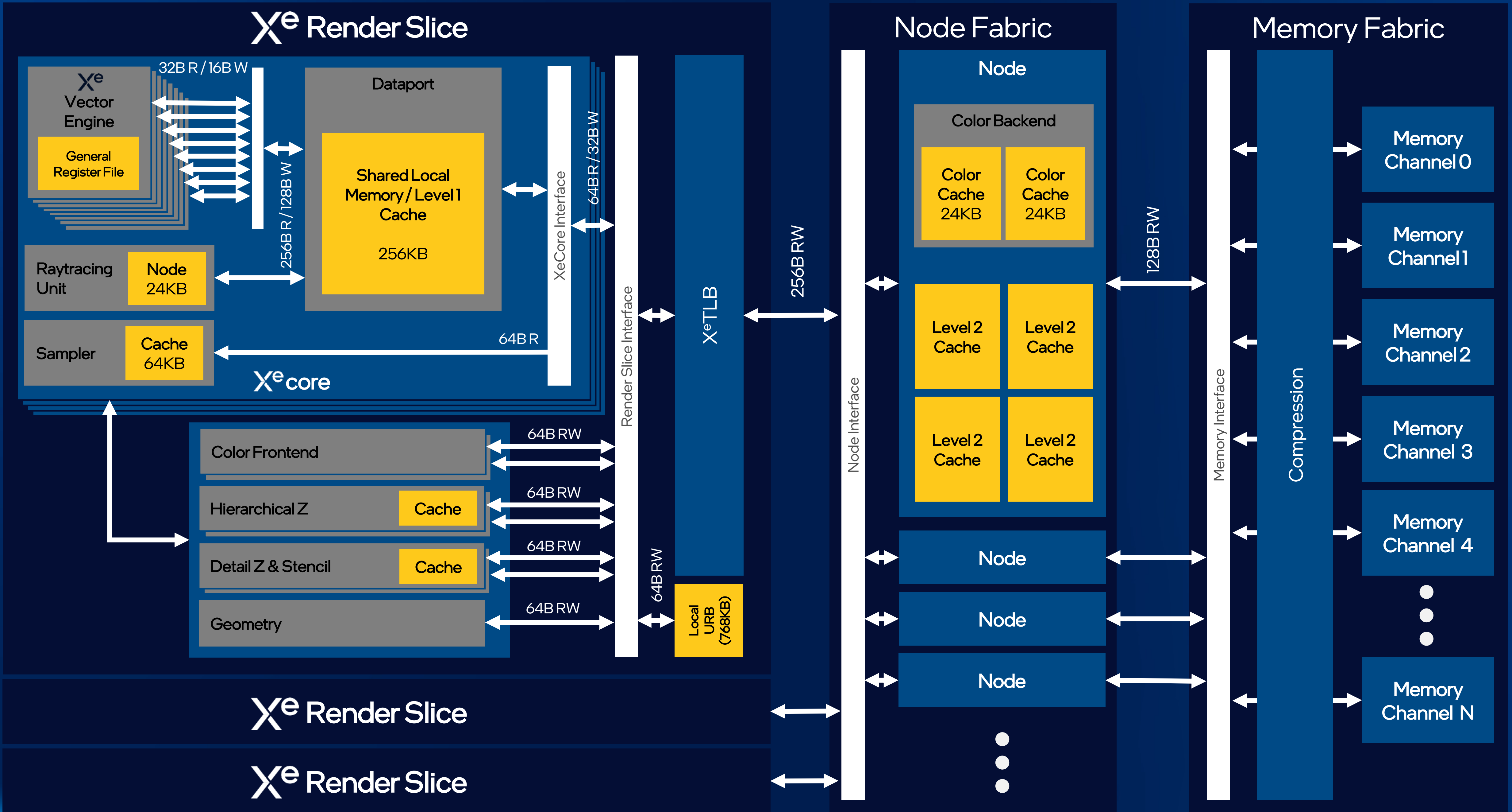
Feedback always welcome!

Minimize data transformation energy.

Thread occupancy is less useful than XVE utilization, but more useful than a sandwich.

Notices & Disclaimers

The end.



Xe Render Slice

Xe Vector Engine

General Register File

32B R / 16B W

256B R / 128B W

Dataport

Shared Local Memory / Level 1 Cache

256KB

XeCore Interface

64B R / 32B W

Raytracing Unit

Node 24KB

Sampler

Cache 64KB

64B R

Xe core

Render Slice Interface

XeTLB

Color Frontend

64B RW

Hierarchical Z

Cache

64B RW

Detail Z & Stencil

Cache

64B RW

Geometry

64B RW

64B RW

Local URB (768KB)

256B RW

Xe Render Slice

Xe Render Slice

Node Fabric

Node Interface

Node

Color Backend

Color Cache 24KB

Color Cache 24KB

Level 2 Cache

Level 2 Cache

Level 2 Cache

Level 2 Cache

Node

Node

Node

128B RW

Memory Fabric

Memory Interface

Compression

Memory Channel 0

Memory Channel 1

Memory Channel 2

Memory Channel 3

Memory Channel 4

Memory Channel N

intel