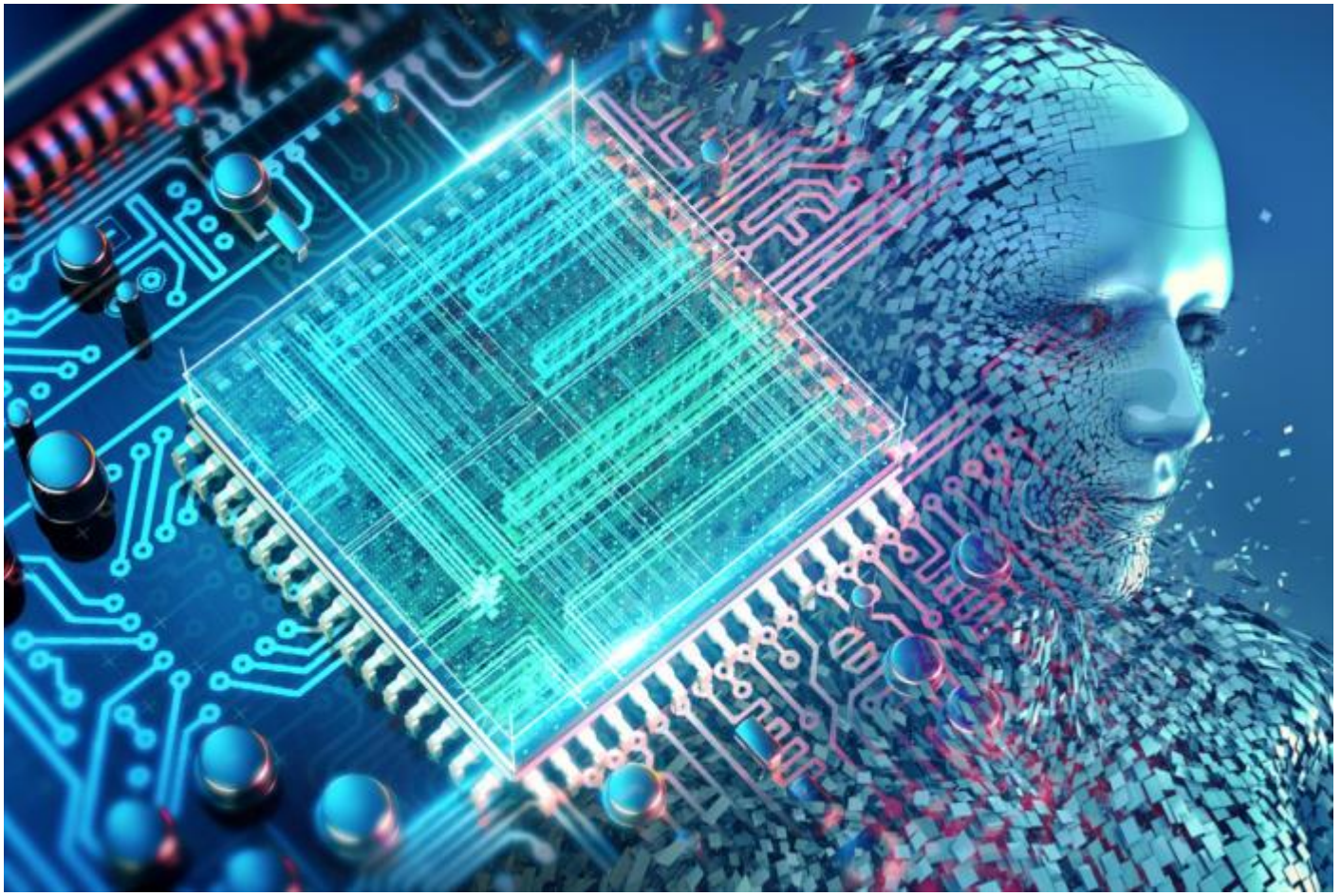
GPU Accelerated Game AI



https://www.networkworld.com/article/3271077/servers/gpus-designed-for-gaming-now-crucial-to-hpc-and-ai.html

Contents

HOCHSCHULE DER MEDIEN

- Motivation
  - Recent Media Articles
  - Nvidia launched their RTX GPUs
- CPU vs GPUs
  - GPU Architecture Basics
  - GPU Programming Model: CUDA
- Game AI on GPUs?
  - Investigating Common AI Techniques
  - Neural Networks and Deep Learning
- Nvidia’s RTX Architecture
  - Real-Time Rendering now relies on AI!?

Motivation (1/5)

HOCHSCHULE DER MEDIEN

- GPUs have evolved much faster than CPUs in recent years
  - Much higher theoretical data throughput
    - Intel i9-9900K: 1,2 TFLOPS
    - RTX 2080 Ti: 14,2 TFLOPS, Tensor-Cores: 113,9 TFLOPS
  - Considerably larger memory bandwidth
    - Intel i9-9900K: max 39.7 GB/s
    - RTX 2080 Ti: 616.0 GB/s
  - Moore's Law is strong with this one!
    - Massively-parallel architecture

- GPUs can be programmed to perform arbitrary tasks
  - Nvidia CUDA, OpenCL, Compute Shaders
  - Successful use in scientific applications
    - Chemistry, physics, finance, weather, numerics, ...
    - Image processing and video coding
    - **Machine Learning and Data Science**
- It's not that easy, though…
  - GPUs, due to their architecture, require specific programming models!

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, http://www.quickmeme.com/meme/33ap, https://www.infoworld.com/article/3299703/deep-learning/what-is-cuda-parallel-programming-for-gpus.html

Motivation (2/5)

HOCHSCHULE DER MEDIEN

- Recent media articles
  - GPUs: Designed for gaming now crucial to HPC and AI (May 2018)
    - "… now powers everything from Adobe Premier and databases to high-performance computing (HPC) and artificial intelligence (AI)."
    - "… up to 5,000 cores... the design lends itself to massive parallel processing."
    - "The GPU is ideally suited to accelerate the processing of SQL queries..."
    - "… 34 of the 50 most popular HPC application packages offer GPU support..."
  - Comparing Hardware for Artificial Intelligence: FPGAs vs. GPUs vs. ASICs (July 2018)
    - "Deep Neural Networks (DNNs)... are all about completing repetitive math algorithms or functions on a massive scale at blazing speeds."
    - "… when the application is also performance-, power-, and latency-critical, FPGAs really shine versus GPUs."
    - "FPGAs are also better than GPUs wherever custom data types exist or irregular parallelism tends to develop."
    - "The architecture of GPUs is not as flexible in making changes to an existing system as are FPGAs."

https://www.networkworld.com/article/3271077/servers/gpus-designed-for-gaming-now-crucial-to-hpc-and-ai.html,
http://eecatalog.com/intel/2018/07/24/comparing-hardware-for-artificial-intelligence-fpgas-vs-gpus-vs-asics/

Motivation (3/5)

HOCHSCHULE DER MEDIEN

- Recent media articles
  - How the GPU became the heart of AI and machine learning (August 2018)
    - "… the big story over the last couple of years, has been the evolution of the GPU from purely the graphics pipeline to really being one of the three core components of deep learning and machine learning."
    - "… new, amazing frameworks that are coming out all the time, the big ones being TensorFlow and Torch and a handful of others."
    - … if you have any access to, say, structured data... a deep-learning model can actually give you a pretty magical amount of predictive power.
    - "Google has re-orientated itself entirely around AI."
    - "You can use the same GPU with videogames as you could use for training deep learning models."
    - "… Nvidia came out with their first GPU that was designed for machine learning, their Volta architecture."

https://www.zdnet.com/article/how-the-gpu-became-the-heart-of-ai-and-machine-learning/

Motivation (4/5)

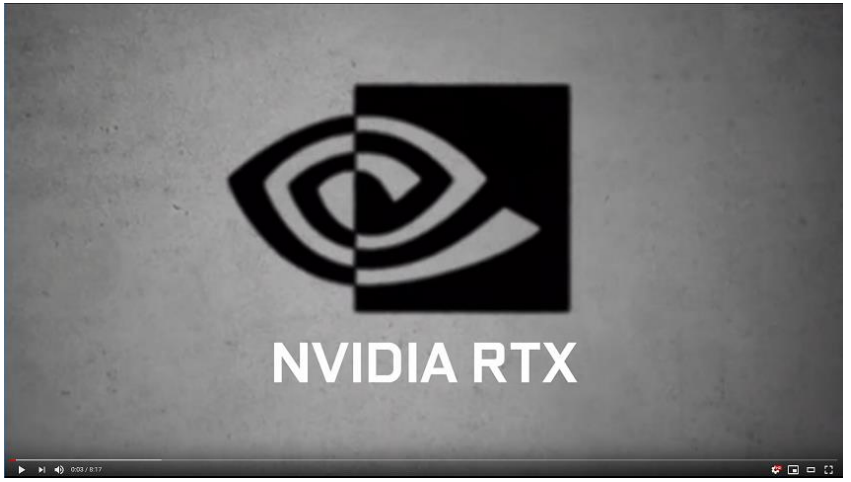
HOCHSCHULE DER MEDIEN

- Recent media articles
  - GPU Usage in Cryptocurrency Mining (February 2018)
    - “Rapidly evolving technology has made cryptocurrency mining a reality on home computers.”
    - “To draw an analogy, the master (CPU) managing the whole organization (the computer system) has a dedicated employee (GPU) to take care of a specialized department (video-rendering functions).”
    - “… the mining process requires higher efficiency in performing similar kinds of repetitive computations.”
    - “The mining device continuously tries to decode the different hashes repeatedly with only one digit changing in each attempt in the data that is getting hashed. A GPU fits better for such dedicated processing.”
    - “Despite technological advancements, certain challenges - like excessive power consumption and limited profit potential – remain which mar the efficiency of the mining hardware.”
    - “GPUs are now facing competition... FPGAs and ASICs, which score better... at performing hash calculations, an essential function to blockchain management in cryptocurrency.”

https://www.investopedia.com/tech/gpu-cryptocurrency-mining/

Motivation (5/5)

- Nvidia launched their new RTX GPUs in August 2018
  - Based on the Turing architecture
  - Tensor cores designed for deep learning
  - Real-time ray tracing acceleration
  - Advancement (more intelligent) shading techniques
- NVIDIA's GPU Technology Conference (GTC) is a global conference series providing training, insights, and direct access to experts on the hottest topics in computing today.

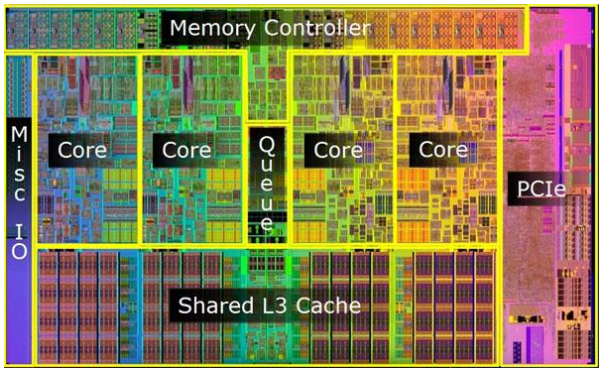


https://www.nvidia.com/en-us/geforce/20-series/, https://devblogs.nvidia.com/nvidia-turing-architecture-in-depth/, https://www.nvidia.com/en-us/gtc/, https://www.youtube.com/watch?v=tjf-1BxpR9c&t=29s

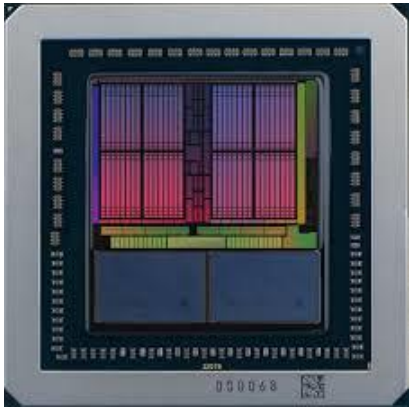
CPU vs GPUs (1/2)

HOCHSCHULE DER MEDIEN

- **CPUs** are designed to execute a single, arbitrary thread as fast as possible
  - Out-of-order execution
  - Branch prediction
  - Memory pre-fetching
  - Multi-level cache hierarchy



- **GPUs** are designed to perform thousands of identical instructions on different data elements in parallel, e.g.:
  - Transforming every single vertex of a mesh to homogeneous clip space
  - Calculating texture and light for millions of pixels on the screen
  - Simulating thousands of independent particles
  - … and so on



https://superuser.com/questions/324284/what-is-meant-by-the-terms-cpu-core-die-and-package,
https://www.tweaktown.com/news/57873/amd-releases-close-up-shot-vega-gpu-die/index.html

CPU vs GPUs (2/2)

https://www.youtube.com/watch?v=-P28LKWTzrI

GPU Architecture Basics (1/6)

- **GPUs operate on many data elements in parallel**
- Example: A diffuse reflectance shader

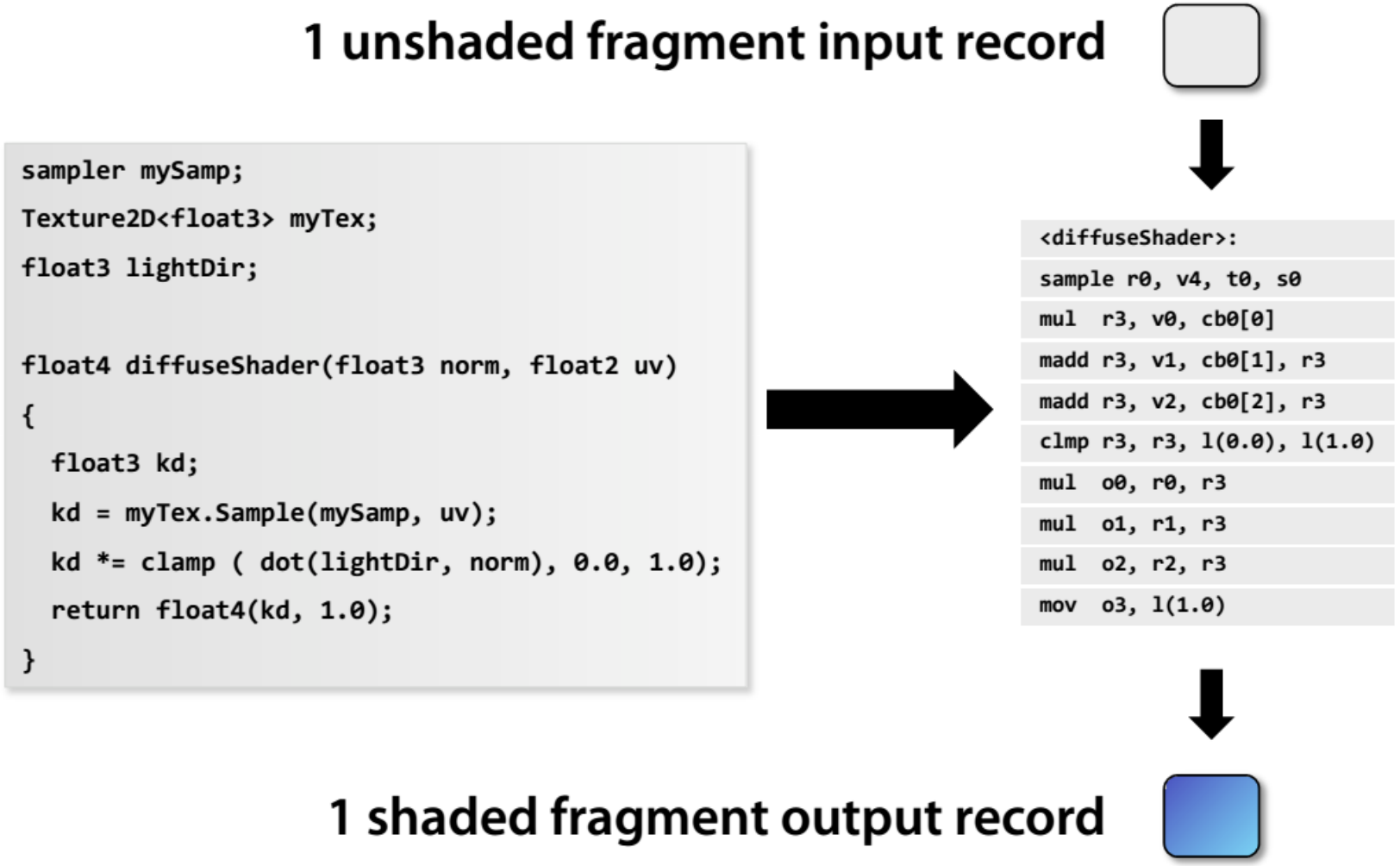


Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

GPU Architecture Basics (2/6)

- **Comparison: CPU style cores**

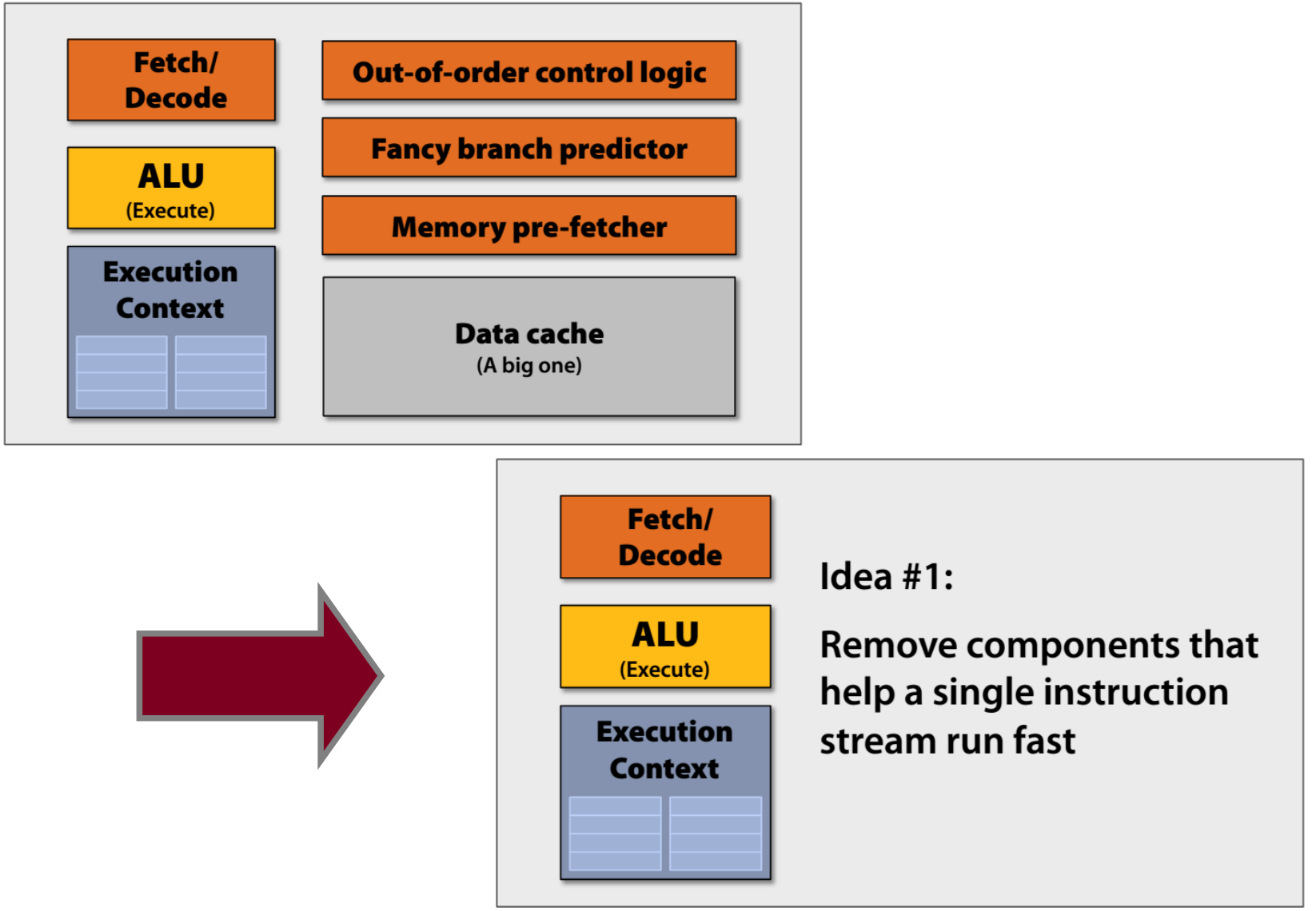
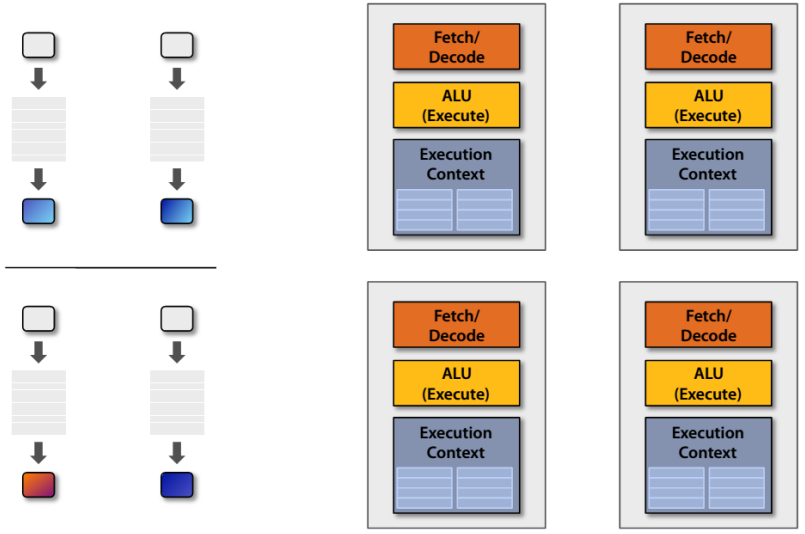


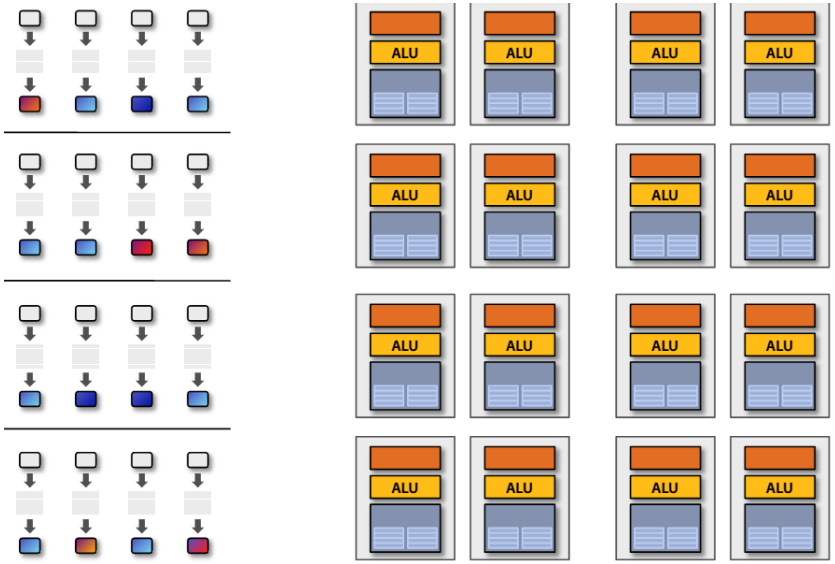
Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

GPU Architecture Basics (3/6)

- **Four cores in parallel**:



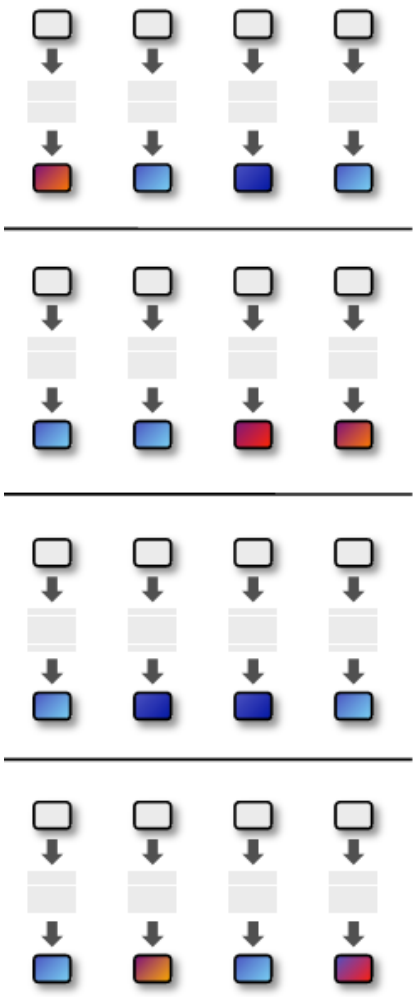
- **Sixteen cores in parallel**:



16 cores = 16 simultaneous instruction streams

Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

GPU Architecture Basics (4/6)

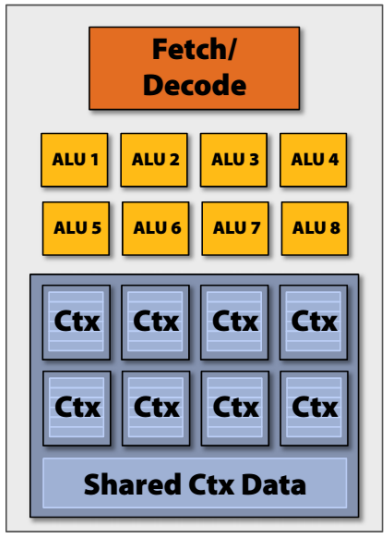


But… many fragments *should* be able to share an instruction stream!

```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

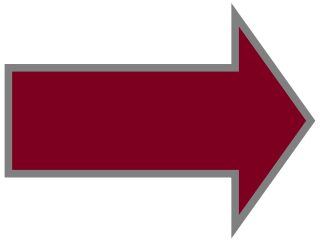
GPU Architecture Basics (5/6)

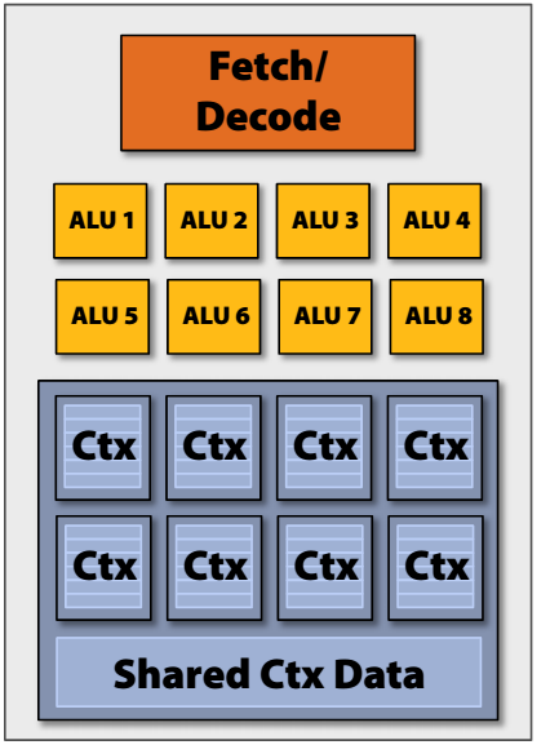


Idea #2:

Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD processing






```
<VEC8_diffuseShader>:
VEC8_sample vec_r0, vec_v4, t0, vec_s0
VEC8_mul  vec_r3, vec_v0, cb0[0]
VEC8_madd vec_r3, vec_v1, cb0[1], vec_r3
VEC8_madd vec_r3, vec_v2, cb0[2], vec_r3
VEC8_clmp vec_r3, vec_r3, l(0.0), l(1.0)
VEC8_mul  vec_o0, vec_r0, vec_r3
VEC8_mul  vec_o1, vec_r1, vec_r3
VEC8_mul  vec_o2, vec_r2, vec_r3
VEC8_mov  vec_o3, l(1.0)
```

New compiled shader:

Processes 8 fragments using vector ops on vector registers

Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

GPU Architecture Basics (6/6)

- **Using 16 of these SIMD cores:**

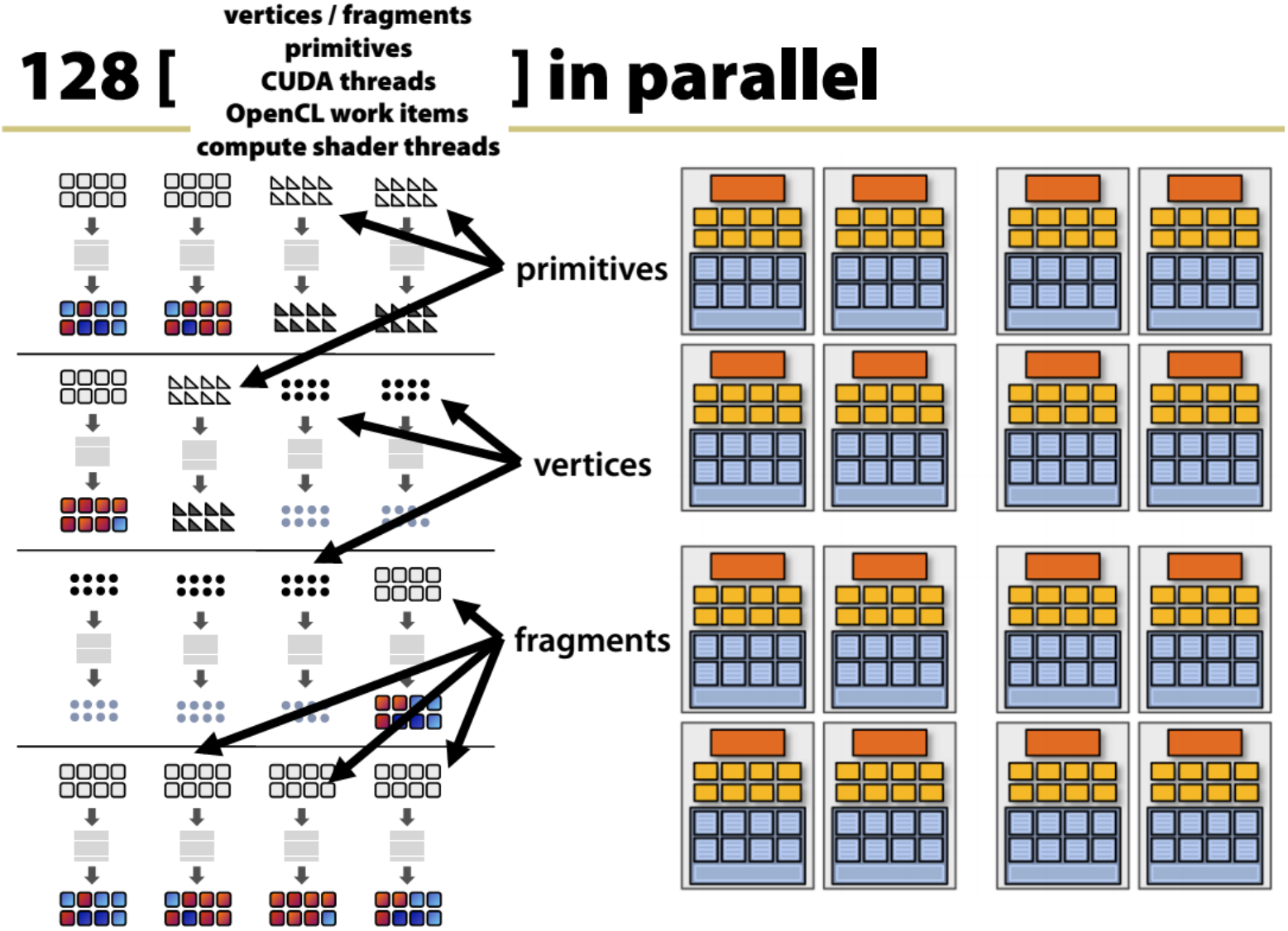
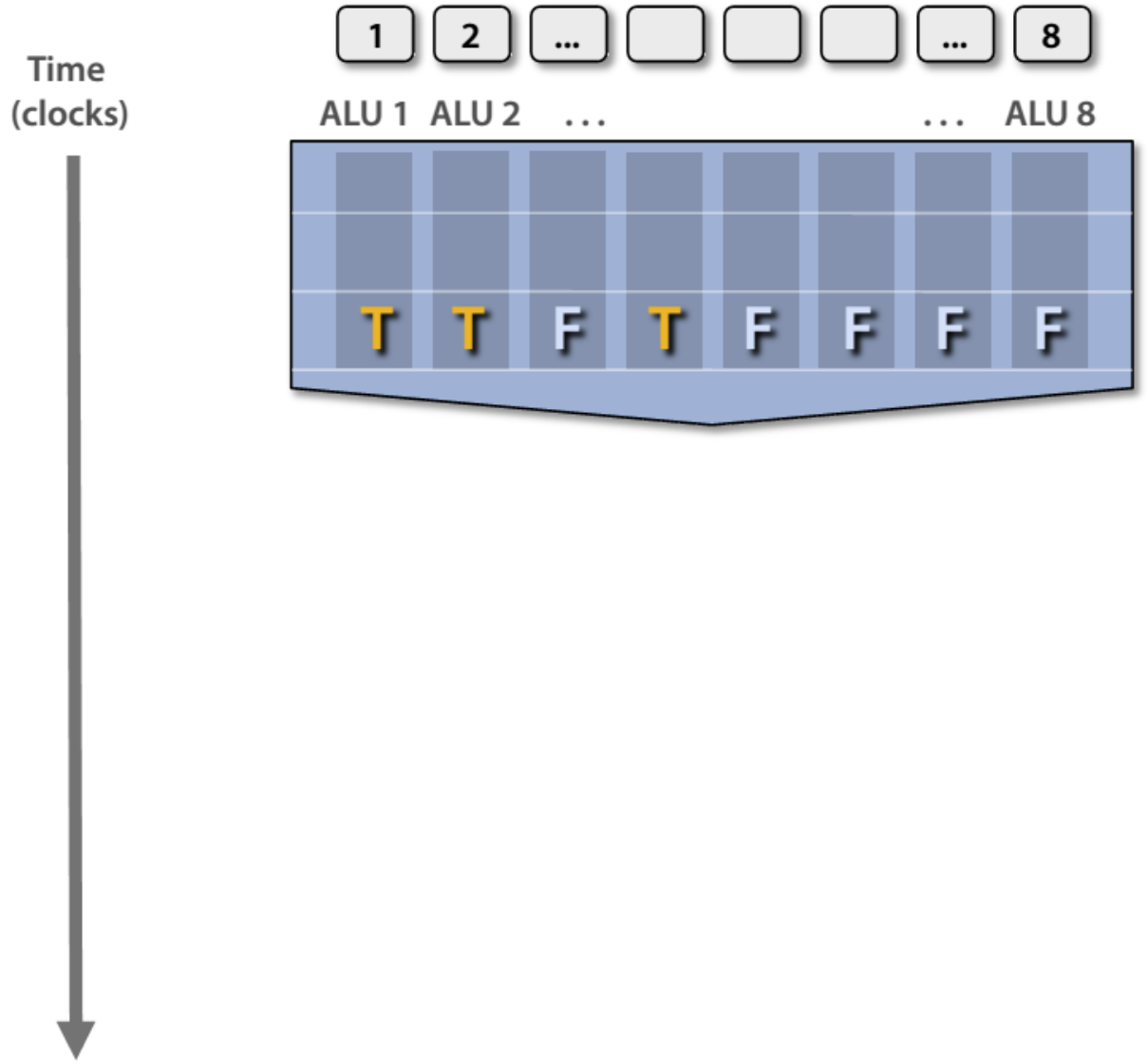


Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

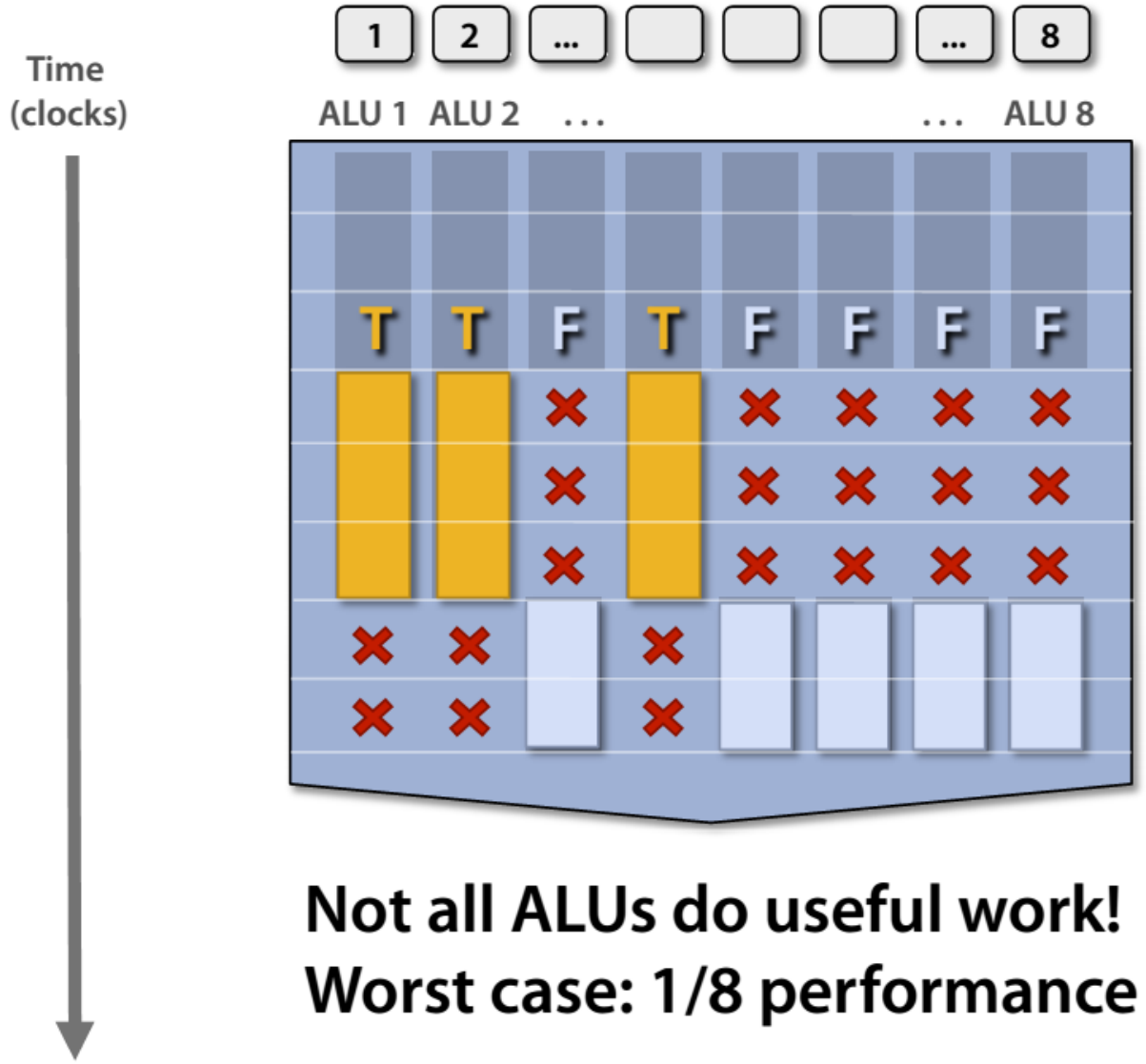
But what about branches? (1/2)




```
<unconditional
shader code>

if (x > 0) {
    y = pow(x, exp);
    y *= Ks;
    refl = y + Ka;
} else {
    x = 0;
    refl = Ka;
}

<resume unconditional
shader code>
```

Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

But what about branches? (2/2)

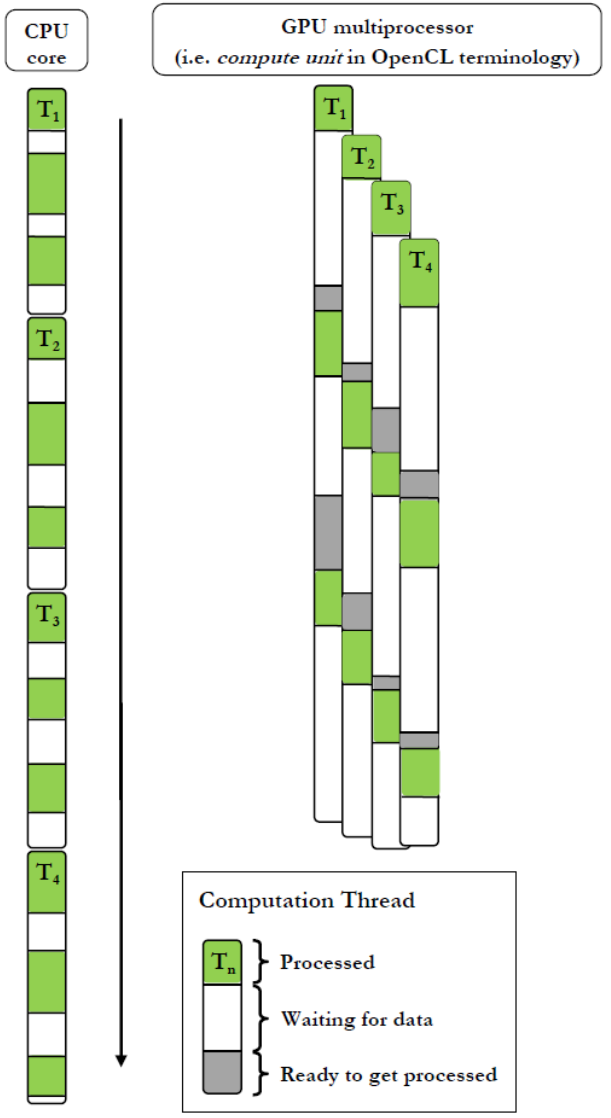



```
<unconditional
shader code>

if (x > 0) {
    y = pow(x, exp);
    y *= Ks;
    refl = y + Ka;
} else {
    x = 0;
    refl = Ka;
}

<resume unconditional
shader code>
```

Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

Dealing with stalls: CPUs vs. GPUs

- Memory access is a bottleneck (latency 200-600 clock cycles)
- CPUs:
  - **avoid** stalls (while waiting for data) by sophisticated caching and flow control
  - Spend lots of transistors for caching and flow control
  - Longer time slices between thread switches (Save and Restore of execution context is expensive)
- GPUs:
  - **hide** stalls by doing computations for other threads in the meantime
  - Spend most of the transistors for compute units
  - frequent thread switches no cost (all execution contexts simultaneously available)



source: OpenCL programming overview (nVidia)

Hiding Shader Stalls (1/2)

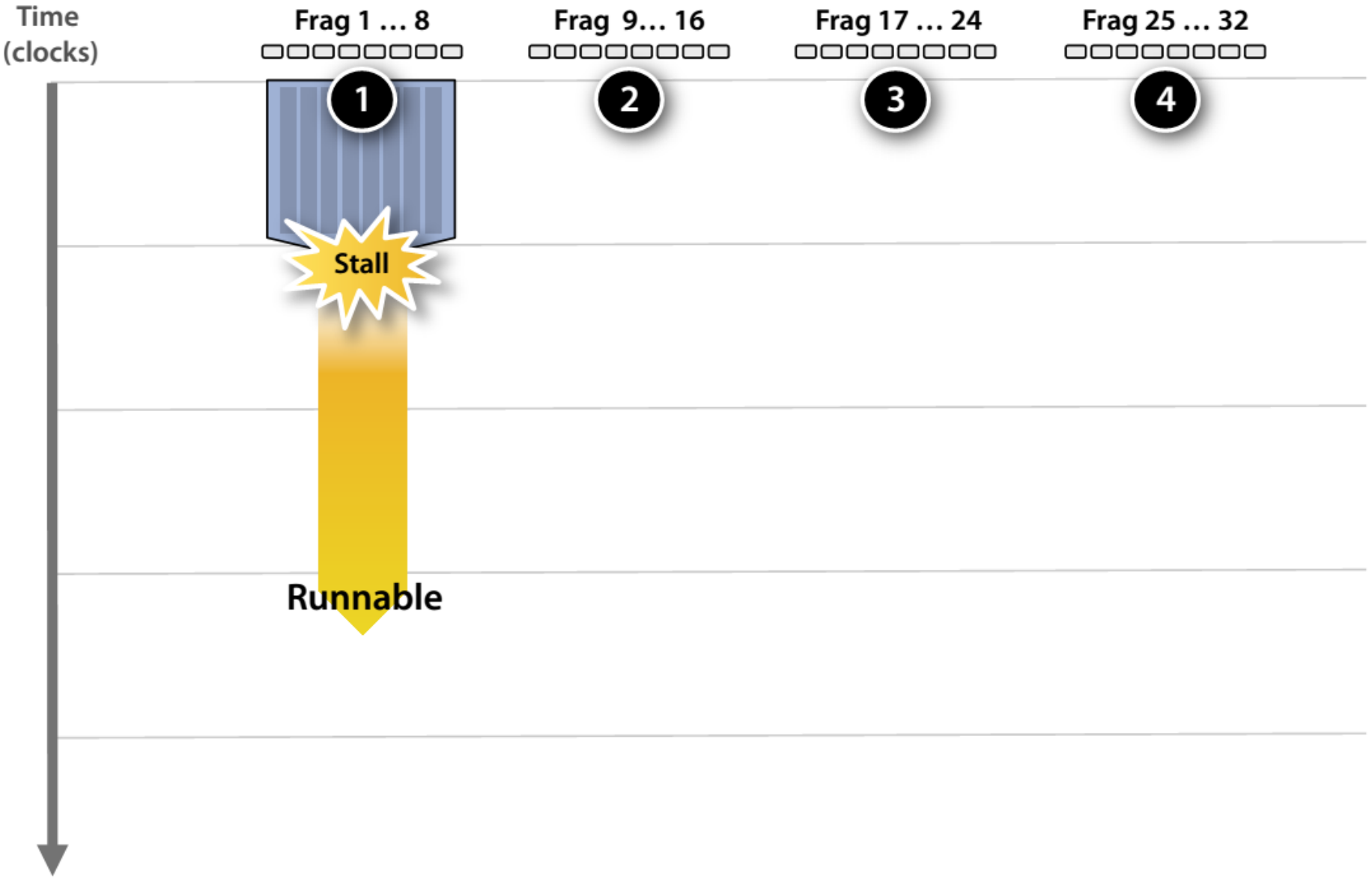


Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

Hiding Shader Stalls (2/2)

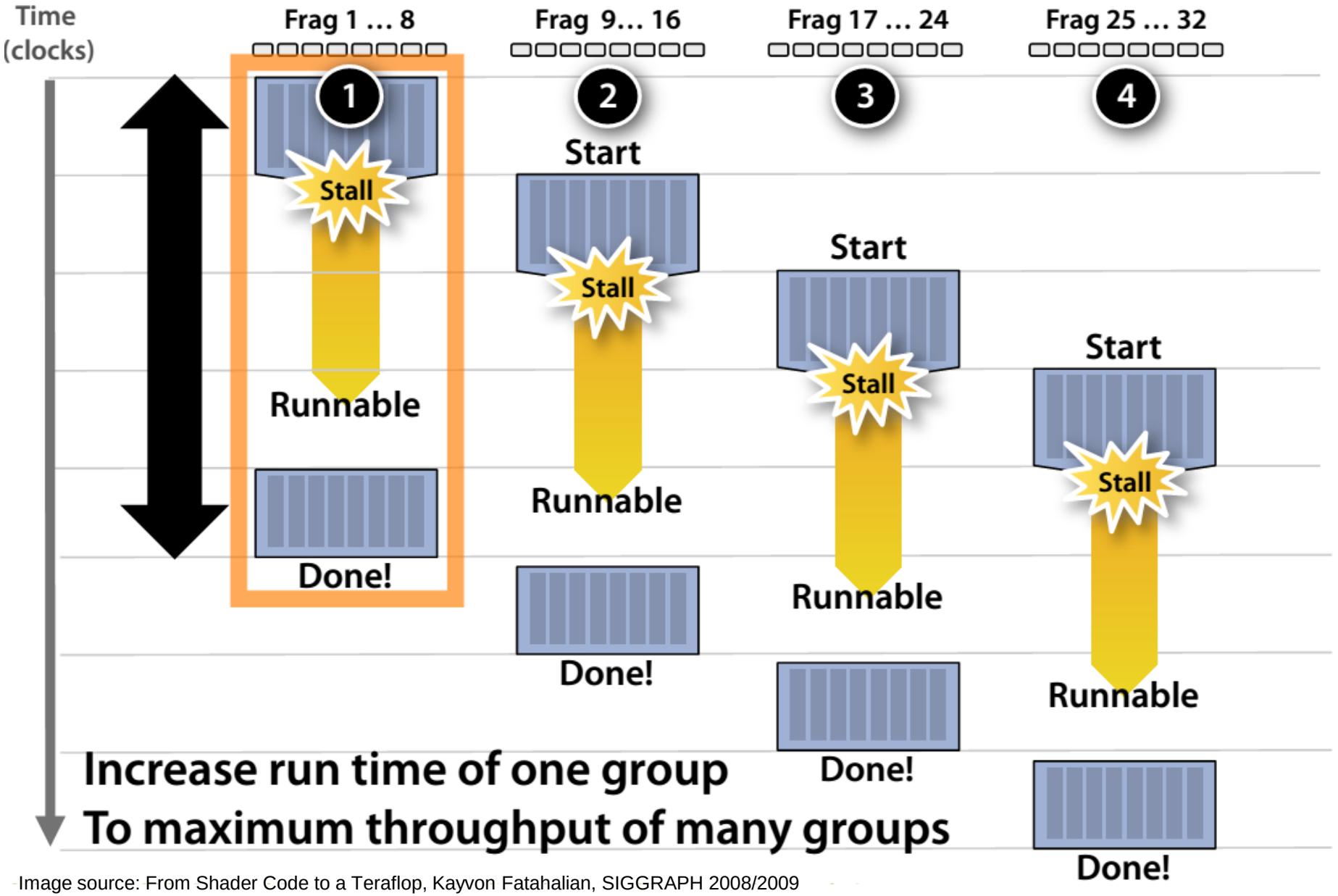


Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

Storing Contexts

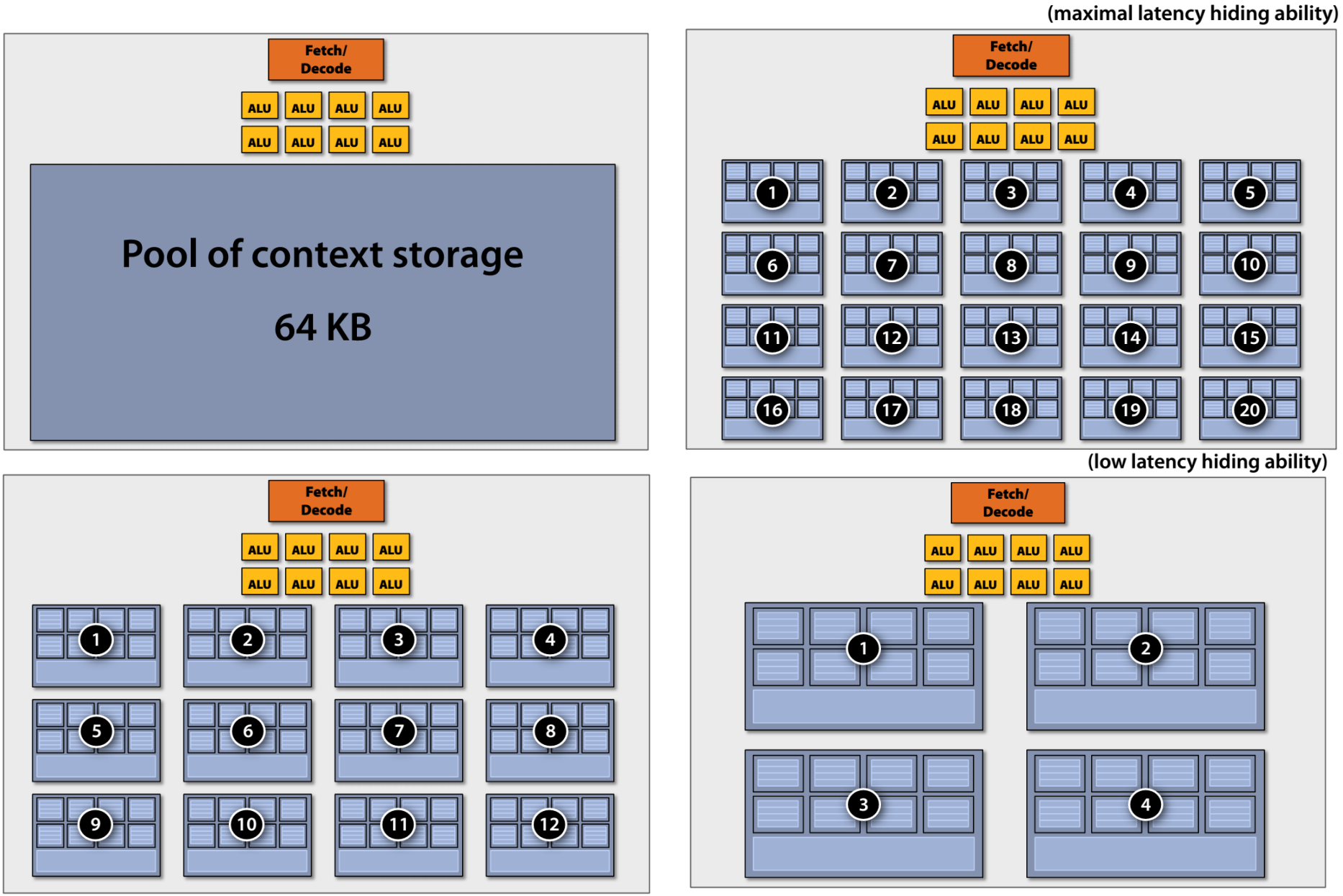
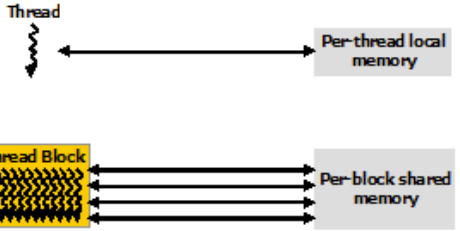


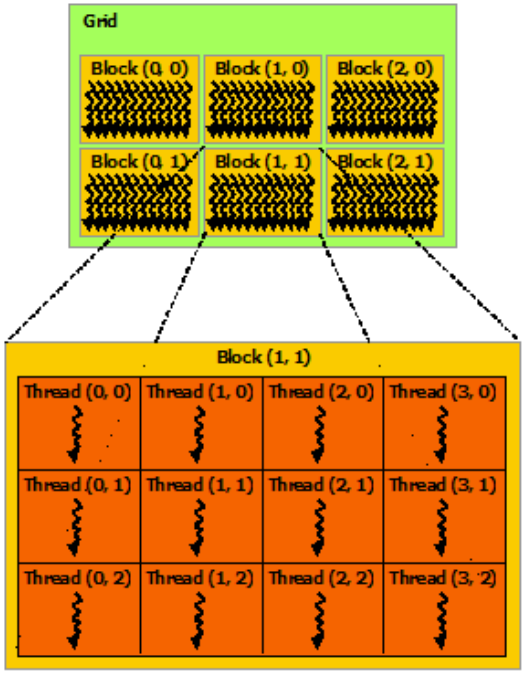
Image source: From Shader Code to a Teraflop, Kayvon Fatahalian, SIGGRAPH 2008/2009

GPU Programming Model: CUDA (1/4)

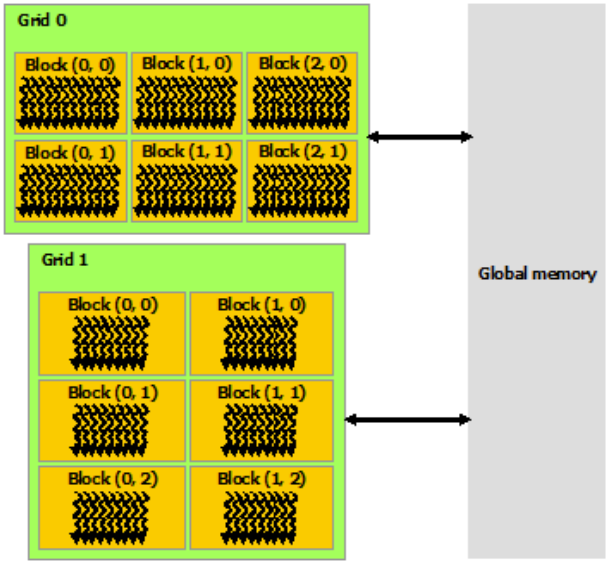
- *Kernels* are C-style functions which are executed N times in parallel by N different CUDA threads

- *Threads* are identified by three-dimensional thread and block *indices*.





- Local, shared and global memory pools

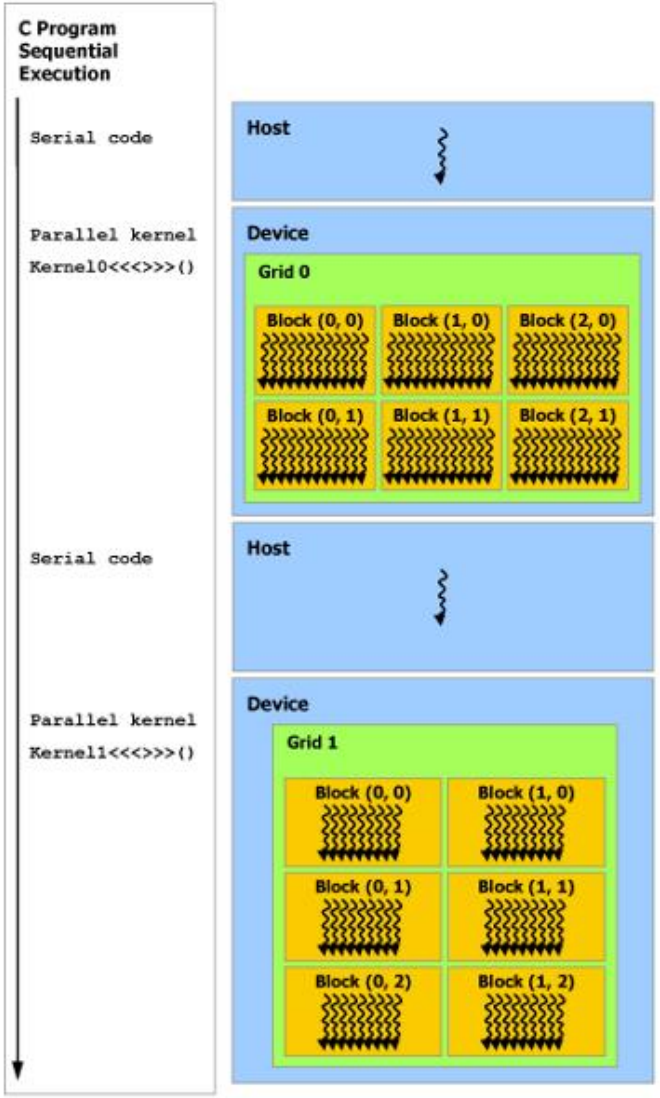


https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

GPU Programming Model: CUDA (2/4)

HOCHSCHULE DER MEDIEN

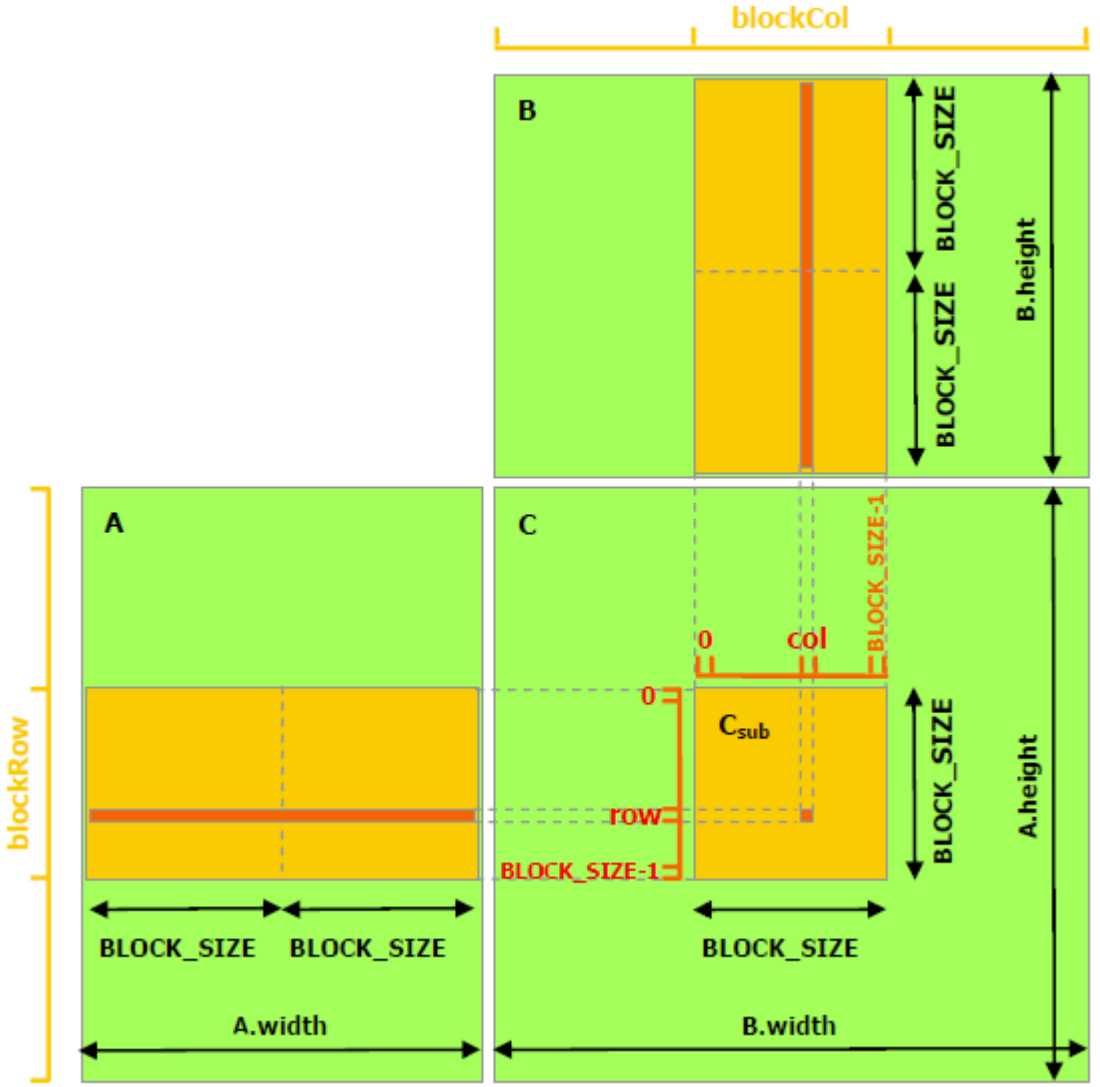
- CUDA threads execute on a physically separate *device* that operates as a coprocessor to the *host* running the main program.
  - Automatic scalability
    - More streaming multiprocessors or CUDA cores
  - Data need to be copied between host and device
    - Synchronous or asynchronous
    - Several mapping options
  - Synchronization must be done explicitly on multiple levels
    - Between Host and Device
    - Between individual kernels
    - Between threads within a kernel
    - Memory fences/barriers within thread groups



https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

GPU Programming Model: CUDA (3/4)

- Example: Matrix multiplication using shared memory



https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

GPU Programming Model: CUDA (4/4)

HOCHSCHULE DER MEDIEN

- Example: Matrix multiplication with shared memory

```
// Matrix multiplication kernel called by MatMul()
 __global__ void MatMulKernel(Matrix A, Matrix B, Matrix C)
{
    // Block row and column
    int blockRow = blockIdx.y; int blockCol = blockIdx.x;
    // Each thread block computes one sub-matrix Csub of C
    Matrix Csub = GetSubMatrix(C, blockRow, blockCol);
    // Each thread computes one element of Csub by accumulating results into Cvalue
    float Cvalue = 0;
    // Thread row and column within Csub
    int row = threadIdx.y; int col = threadIdx.x;
    // Loop over all the sub-matrices of A and B that are required to compute Csub
    // Multiply each pair of sub-matrices together and accumulate the results
    for (int m = 0; m < (A.width / BLOCK_SIZE); ++m) {
        // Get sub-matrix Asub of A
        Matrix Asub = GetSubMatrix(A, blockRow, m);
        // Get sub-matrix Bsub of B
        Matrix Bsub = GetSubMatrix(B, m, blockCol);
        // Shared memory used to store Asub and Bsub respectively
        __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];
        __shared__ float Bs[BLOCK_SIZE][BLOCK_SIZE];
        // Load Asub and Bsub from device memory to shared memory
        // Each thread loads one element of each sub-matrix
        As[row][col] = GetElement(Asub, row, col); Bs[row][col] = GetElement(Bsub, row, col);
        // Synchronize to make sure the sub-matrices are loaded before starting the computation
        __syncthreads();
        // Multiply Asub and Bsub together
        for (int e = 0; e < BLOCK_SIZE; ++e) Cvalue += As[row][e] * Bs[e][col];
        // Synchronize to make sure that the preceding computation is done before loading two new
        // sub-matrices of A and B in the next iteration
        __syncthreads();
    }
    // Write Csub to device memory Each thread writes one element
    SetElement(Csub, row, col, Cvalue);
}
```

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Game AI on GPUs? (1/7)

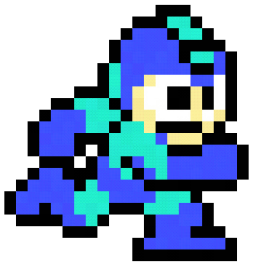
- The AI in most modern games addresses three basic needs:
  1. The ability to move characters
  2. The ability to make decisions about where to move
  3. The ability to think tactically or strategicall

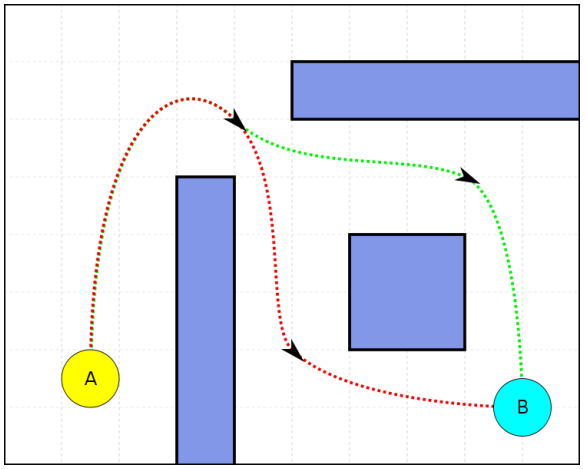
- **Pac Man** (1979) used a state machine for each enemy
  - Either chasing you or running away
  - They took a semi-random route at each junction
- **Goldeneye 007** (1997) used a sense simulation system
  - A character could see their colleagues and would notice if they were killed
- **Creatures** (1997) has one of the most complex AI systems seen in a game
  - A neural network-based brain for each creature

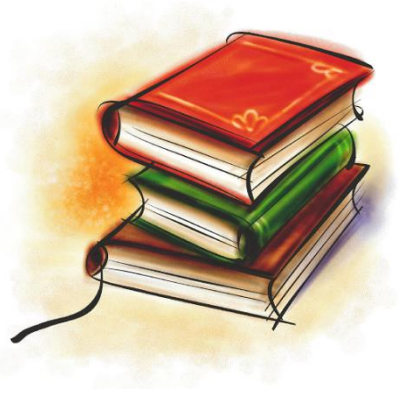
http://www.retrogameage.com/shop/nintendo-64/goldeneye-007-nintendo-64/, https://www.myabandonware.com/game/creatures-3i7

Game AI on GPUs? (2/7)

- Investigating common AI techniques
  - Movement
  - Pathfinding
  - Decision Making
  - Learning



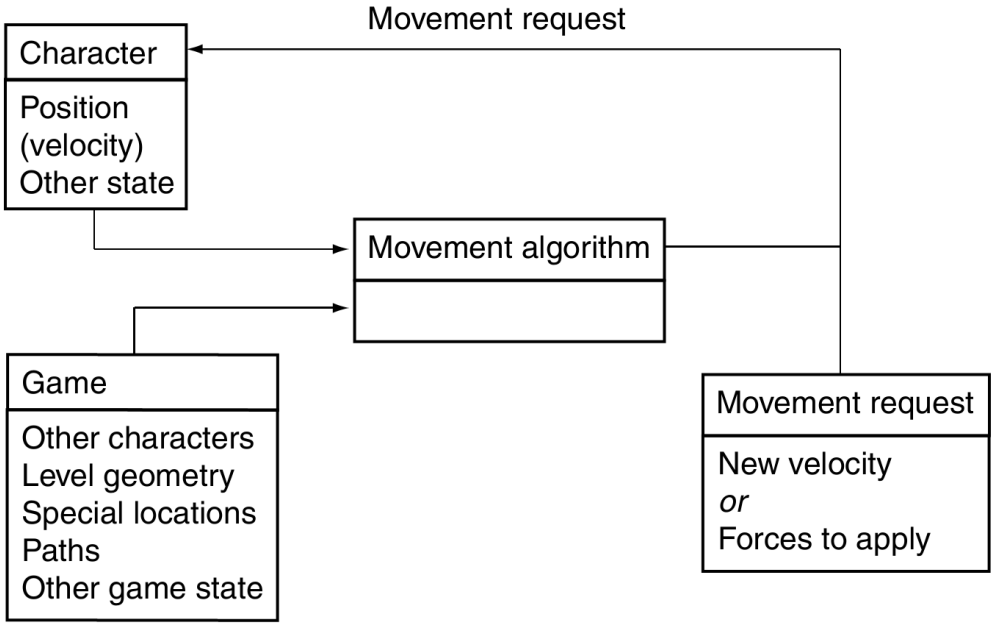




https://www.pinterest.com/pin/511299363917995246/, https://en.wikipedia.org/wiki/Pathfinding, http://darksouls.wikia.com/wiki/File:Thinking-smiley.jpg, http://hddfhm.com/clip-art/books-cliparts.html, https://www.123rf.com/stock-photo/playbook.html?sti=lzqynygzfabh64awdz|

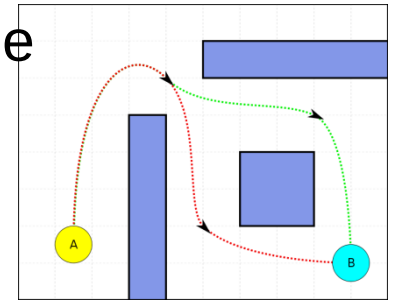
Game AI on GPUs? (3/7)

- **Movement**: One of the most fundamental requirements of AI is to move characters around in the game sensibly.

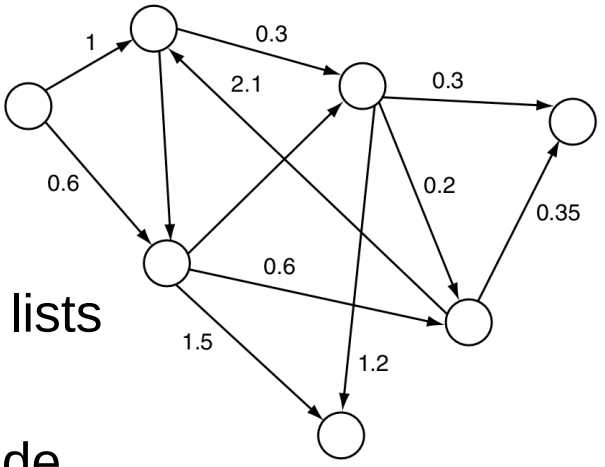


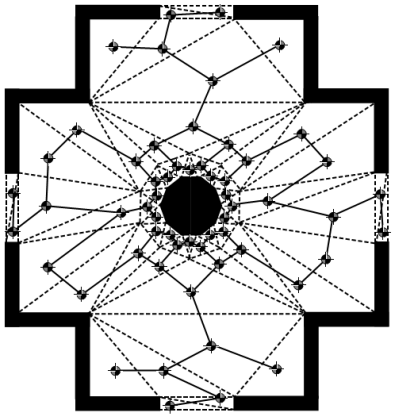
- Suitable for GPUs?
  - Can only be parallelized on a per-agent level
  - Interaction between agents would be very problematic
  - Terrain / map / navmesh data would need to be “flattened”
  - Output would always need to be send back to the host

Game AI on GPUs? (4/7)

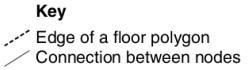


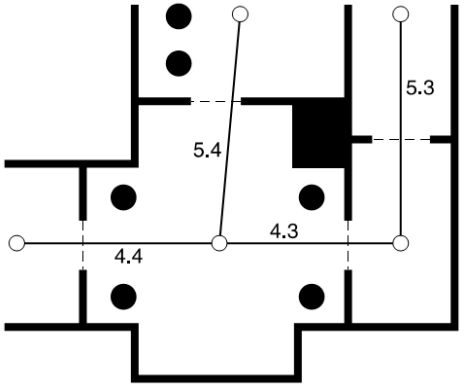
- **Pathfinding**: The AI must be able to calculate a suitable route through the game level to get from where it is now to its goal.
  - Many games rely on the A* algorithm
    - It requires the game level be represented as a directed negative weighted graph.





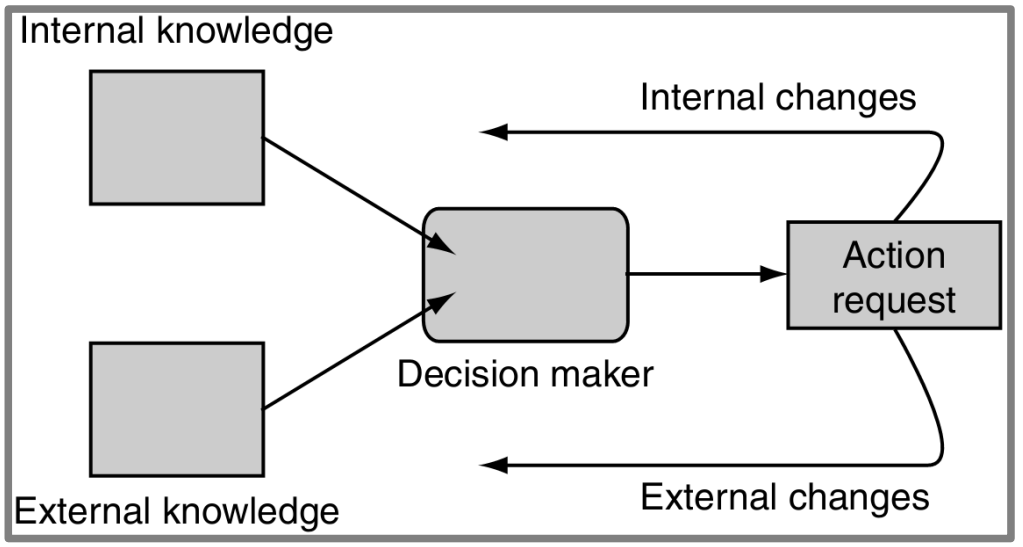
- Suitable for GPUs?
  - Management of graphs and node lists be highly inefficient
  - Way too many branches in the code
  - Some research exists, e.g.:
    - Zhou and Zeng, “Massively Parallel A* Search on a GPU”, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015
    - Bleiweiss, “GPU Accelerated Pathfinding”, NVIDIA, 2008





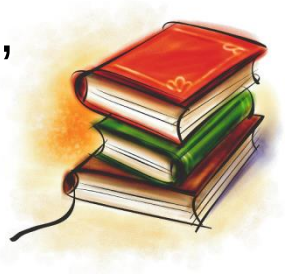
Game AI on GPUs? (5/7)

- **Decision Making**: The ability of a character to decide what to do
  - Decision Trees
  - Finite State Machines
  - Behaviour Trees



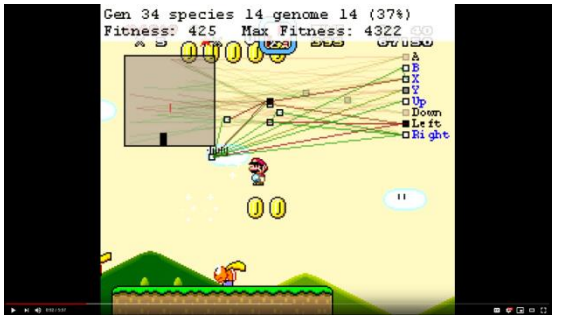
- Suitable for GPUs?
  - Can only be parallelized on a per-agent level
  - Internal and external knowledge is hard to organize, acquire and update
  - By nature requires many branches (decisions), thus would run very inefficiently on GPUs
  - Output would always need to be send back to the host

Game AI on GPUs? (6/7)



- **Learning**: In principle has the potential to adapt to the player, learning their tricks and providing a consistent challenge.
  - **Online learning**
    - Performed during the game, while the player is playing.
    - Allows characters to adapt dynamically to the player's style.
  - **Offline learning**
    - The majority of learning in game AI is done offline, either between levels of the game or more often at the development studio
    - This allows more unpredictable learning algorithms to be tried out and their results to be tested exhaustively.
    - The learning algorithms in games are usually applied **offline**; it is rare to find games that use any kind of **online learning**.
- Suitable for GPUs?
  - Many combinations can be tested in parallel
  - Training of artificial neural networks works very well
    - The trained network data can be stored and shipped with the game

Game AI on GPUs? (7/7)

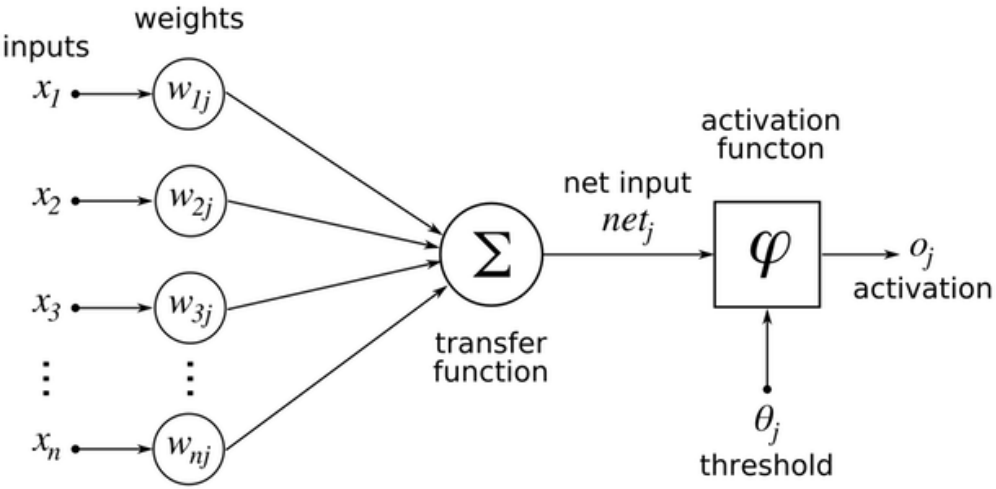


- Machine Learning: An algorithm plays Super Mario World
  - Artificial Neural Network
  - Genetic Algorithm
- Several other such experiments exist:
  - https://www.activestate.com/blog/building-game-ai-using-machine-learning-working-tensorflow-keras-and-intel-mkl-python/
    - "It's remarkable how consistent the behaviour of this simple network is..."
  - https://medium.com/acing-ai/how-i-build-an-ai-to-play-dino-run-e37f37bdf153
    - "I trained my model for around 2 million frames for a week."
  - https://towardsdatascience.com/building-a-deep-neural-network-to-play-fifa-18-dce54d45e675?gi=9a4fd5113a89
    - "I believe it can get very close to human level performance with many more hours of training data, ..."
- But… what about AAA games!?

https://www.youtube.com/watch?v=qv6UVOQ0F44

Neural Networks and Deep Learning (1/4)

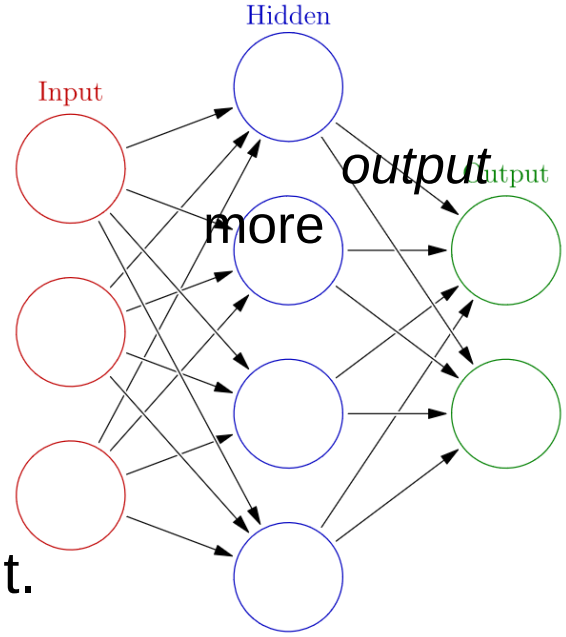
- The inner-workings of the human brain are often modeled around the concept of *Neurons* (~100 billion) and the networks of neurons known as *Biological Neural Networks*.
  - Neurons communicate with one another across their *synapses*.
  - *Activation*: A single neuron will pass a message to another neuron across if the sum of weighted *input signals* from one or more neurons (summation) into it is great enough to cause the *message transmission*.
  - Each neuron applies a function or *transformation* to the weighted inputs.
  - The *thinking* or processing that our brain carries out are the result of these neural networks in action.
- *Artificial Neural Networks* (ANNs) are statistical models directly inspired by, and partially modelled on biological neural networks.



https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html

Neural Networks and Deep Learning (2/4)

- Artificial Neural Networks
  - Are capable of modelling and processing nonlinear relationships between inputs and outputs *in parallel*.
  - They contain *adaptive weights* along paths between neurons that can be tuned by a *learning algorithm* that learns from observed data in order to improve the model.
  - The *cost function* is what's used to learn the optimal solution to the problem being solved.
  - In a simple model, the first layer is the *input layer*, followed by one *hidden layer*, and lastly by an *output layer*. Each layer can contain one or more neurons.



- **Deep Learning**, describes certain types of ANNs and related algorithms. They process this data through many layers of nonlinear transformations of the input data in order to calculate a target output.

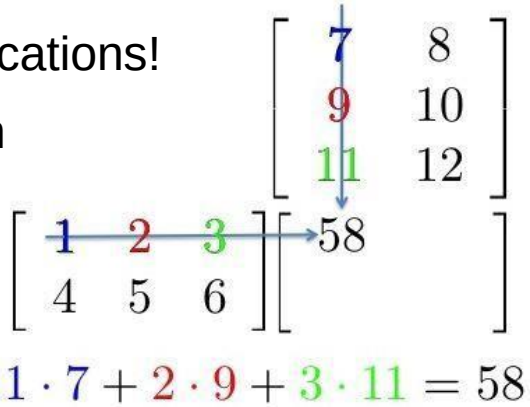
Neural Networks and Deep Learning (3/4)

HOCHSCHULE DER MEDIEN

- Deep Learning
  - Falls under the group of techniques known as *feature learning* or *representation learning*.
  - The machine learning algorithms themselves *learn* the optimal parameters to create the best performing model.
  - For neural network-based deep learning models, the number of layers are greater than in so-called *shallow* learning algorithms.
  - Deep learning algorithms rely more on optimal model selection and optimization through model tuning.
  - In addition to statistical techniques, neural networks and deep learning leverage concepts and techniques from signal processing as well, including nonlinear processing and/or transformations.
- Training deep neural networks can take a significant amount of time.
  - Lots of data. High complexity.
  - GPUs can be leveraged to drastically accelerate the learning process.

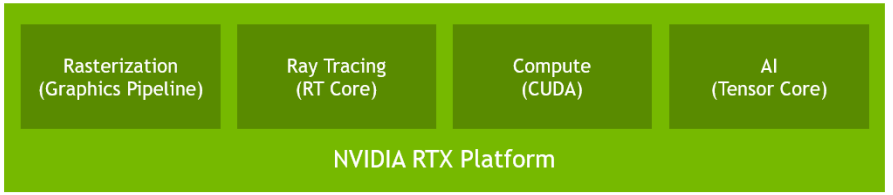
Neural Networks and Deep Learning (4/4)

- Deep Learning with GPUs
  - GPU acceleration has become critical for deep learning as models have become more complex and datasets have grown in size.
  - GPUs were recognized early on as having an architecture well-adapted to speeding up the massively parallel array calculations at the heart of deep learning.
  - In a deep learning model, in *forward pass*, input is passed through the neural network and after processing the input, an output is generated. Whereas in *backward pass*, we update the weights of neural network on the basis of error we get in forward pass.
    - Both of these operations are essentially matrix multiplications!
    - Each element in one row of first array is multiplied with one column of second array.
    - First array can be considered as input to the ANN.
    - Second array can be considered as weights of ANN.



https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/,
https://www.youtube.com/watch?v=R_QQETptCAU

Nvidia RTX (1/4)



- The RTX platform combines…
  - Ray Tracing
    - Calculates the colour of pixels by tracing the path that light would take if it were to travel from the eye of the viewer through the virtual 3D scene.
    - GPUs include dedicated ray tracing acceleration hardware.
      - Space- and mesh-subdivision for accelerated path tracing.
  - Artificial Intelligence
    - AI-based features that accelerate and enhance graphics, photos imaging and video processing.
    - Tensor Cores to maximize the efficiency of these operations.
  - Rasterization
    - Mesh Shading: new shader model for the vertex, tessellation, and geometry.
    - Variable Rate Shading: allows to control shading rate dynamically.
    - Texture-Space Shading
    - Multi-View Rendering

https://developer.nvidia.com/rtx

Nvidia RTX (2/4)

- Turing GPU Overview

INTRODUCING TURING

TU102 – FULL CONFIG

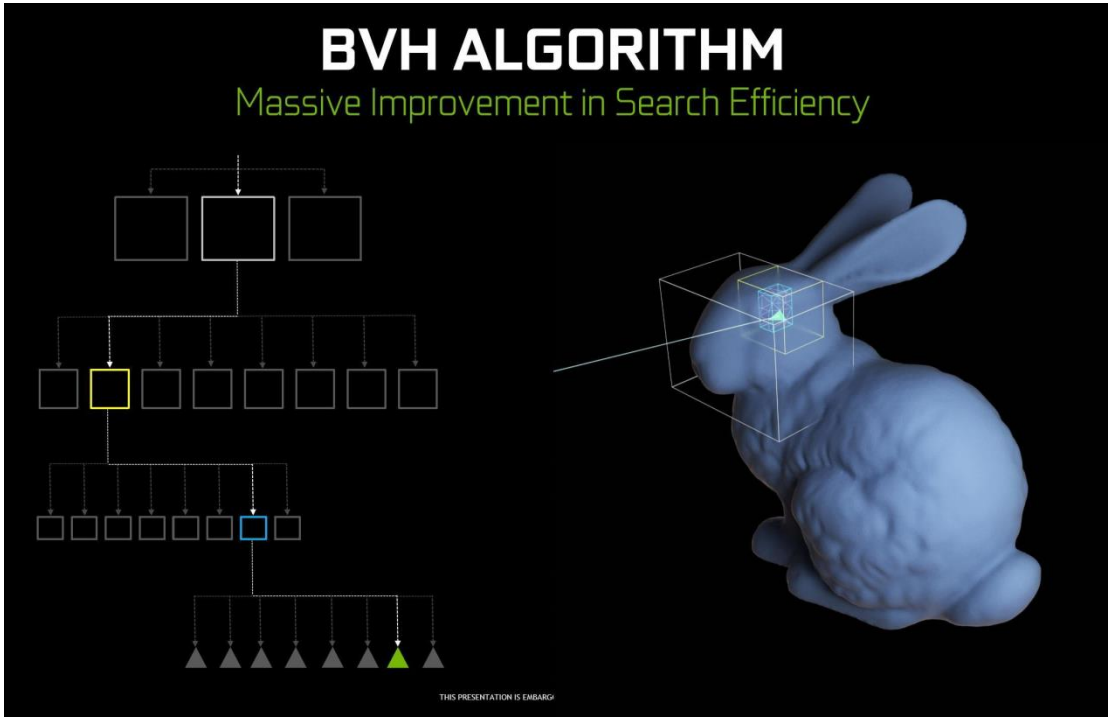
18.6 BILLION TRANSISTORS

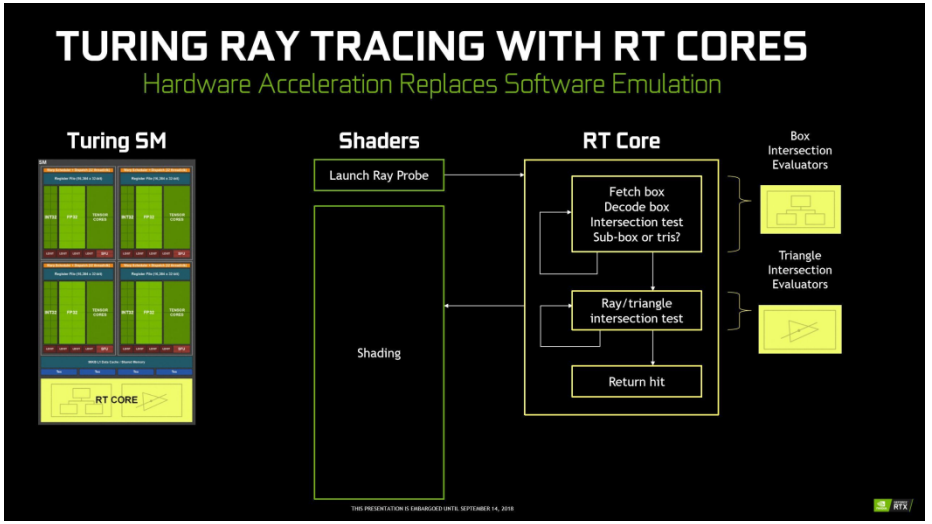
SM	72
CUDA CORES	4608
TENSOR CORES	576
RT CORES	72
GEOMETRY UNITS	36
TEXTURE UNITS	288
ROP UNITS	96
MEMORY	384-bit 7 GHz GDDR6
NVLINK CHANNELS	2

THIS PRESENTATION IS EMBARGOED UNTIL SEPTEMBER 14, 2018

https://www.pcworld.com/article/3305717/components-graphics/nvidia-turing-gpu-geforce-rtx-2080-ti.html

Nvidia RTX (3/4)

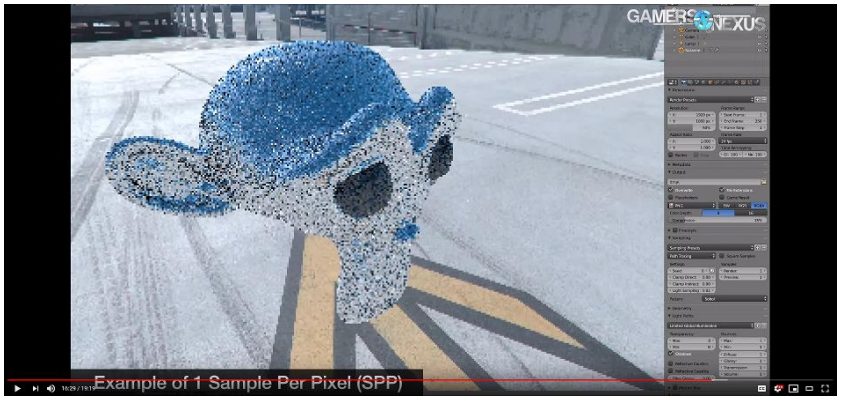




Nvidia RTX (4/4)

- Real-Time Rendering now relies on AI!?
- NGX (Neural Graphics Framework) utilizes deep neural networks (DNNs) and set of ‘Neural Services’ to perform AI-based functions that accelerate and enhance graphics, rendering, and other client-side applications,”
  - Ray Tracing with deep-learned denoise algorithms

3:00 – 6:00



16:00

  - Deep Learning Super Sampling (DLSS), is an antialiasing method that provides sharper results at a much lower performance cost.
  - Other shader effects can be learned as well! E.g. AO, Motion Blur, etc.

https://www.youtube.com/watch?v=lFnWy0Odsh8&t=26s, https://www.youtube.com/watch?v=jaUP4LucmZM

Deep Learned Denoising Example

https://www.youtube.com/watch?v=l-5NVNgT70U

DLSS Example 1

https://towardsdatascience.com/deep-learning-based-super-resolution-without-using-a-gan-11c9bb5b6cd5

DLSS Example 2

https://towardsdatascience.com/deep-learning-based-super-resolution-without-using-a-gan-11c9bb5b6cd5

Conclusions

- Machine Learning is already part of our everyday life
  - Photo and video recognition
  - Natural language processing
  - … and many more!
- Massively parallel GPUs power lots of these learning algorithms
- New GPUs feature dedicated learning hardware
  - Games don’t utilize this yet!
- Real-time rendering is one advertised use-case
  - Denoising
  - Anti Aliasing
  - Ambient Occlusion
  - … more in the future.