Big Data Course

Chap 5-
Cache and GPU

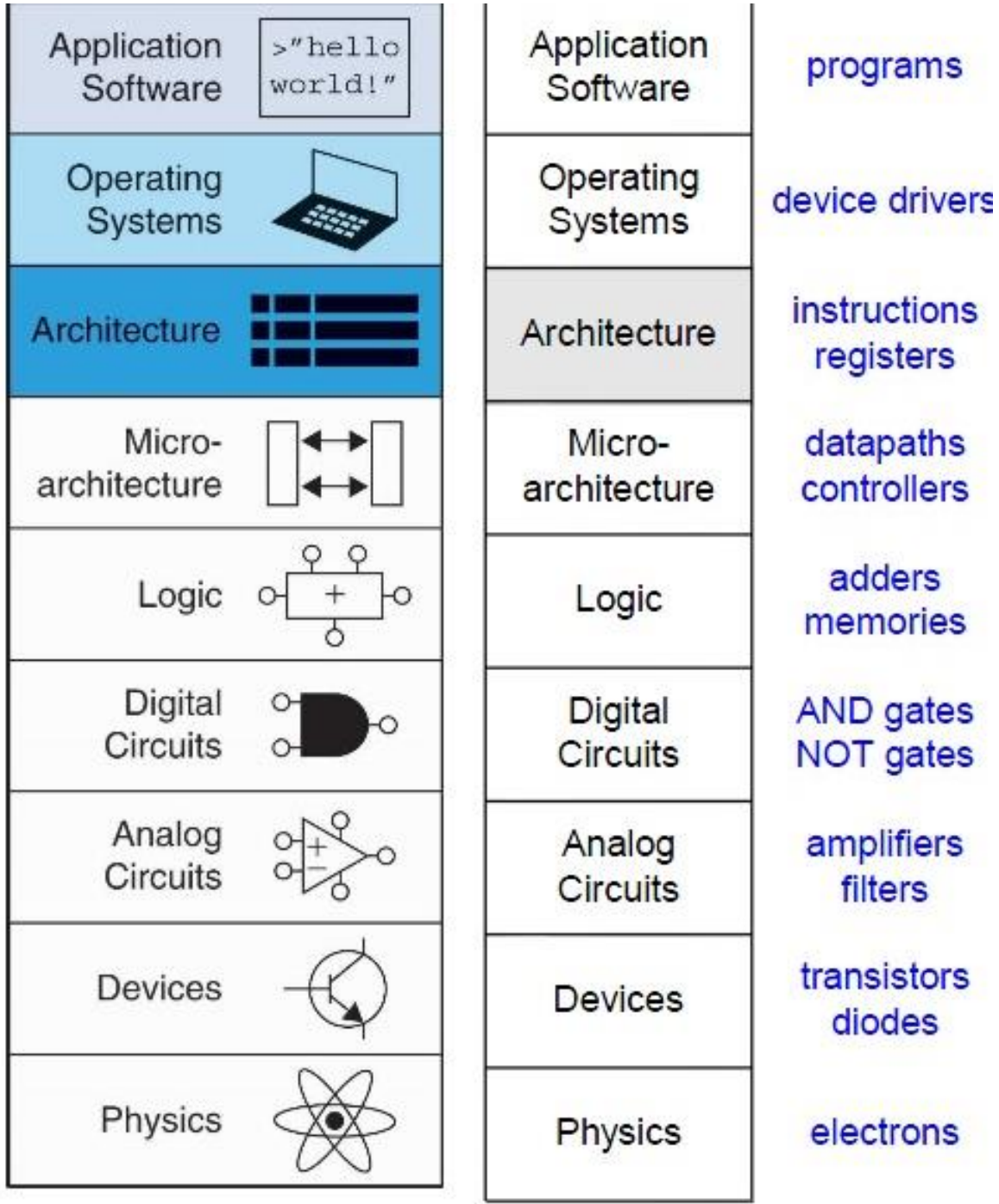
Electrical Engineering department of

Amirkabir University of technology

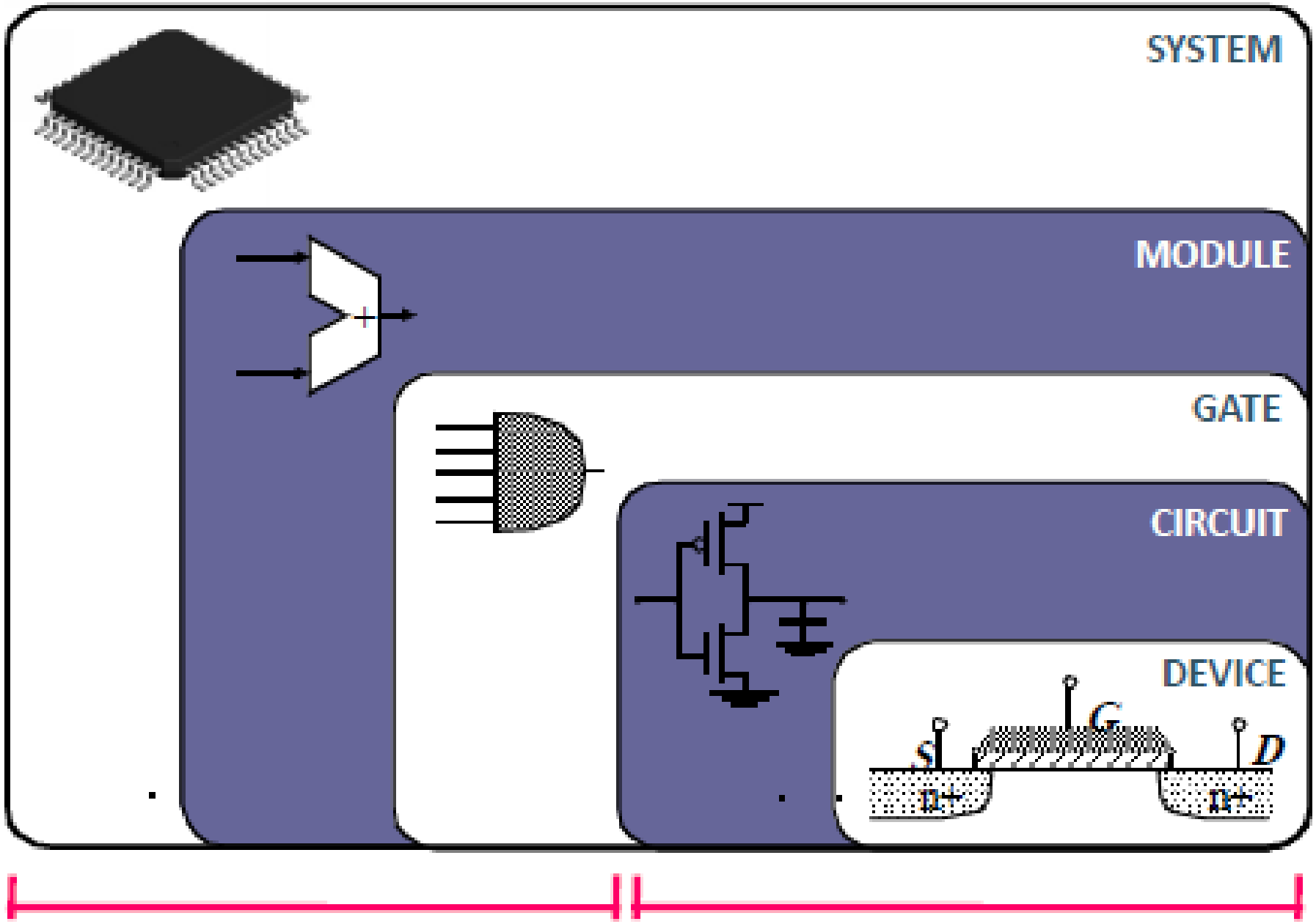
Dr. Mohammadreza Pourfard

August 2025

- **Architecture:** programmer's view of computer
  - Defined by instructions & operand locations
- **Microarchitecture:** how to implement an architecture in hardware



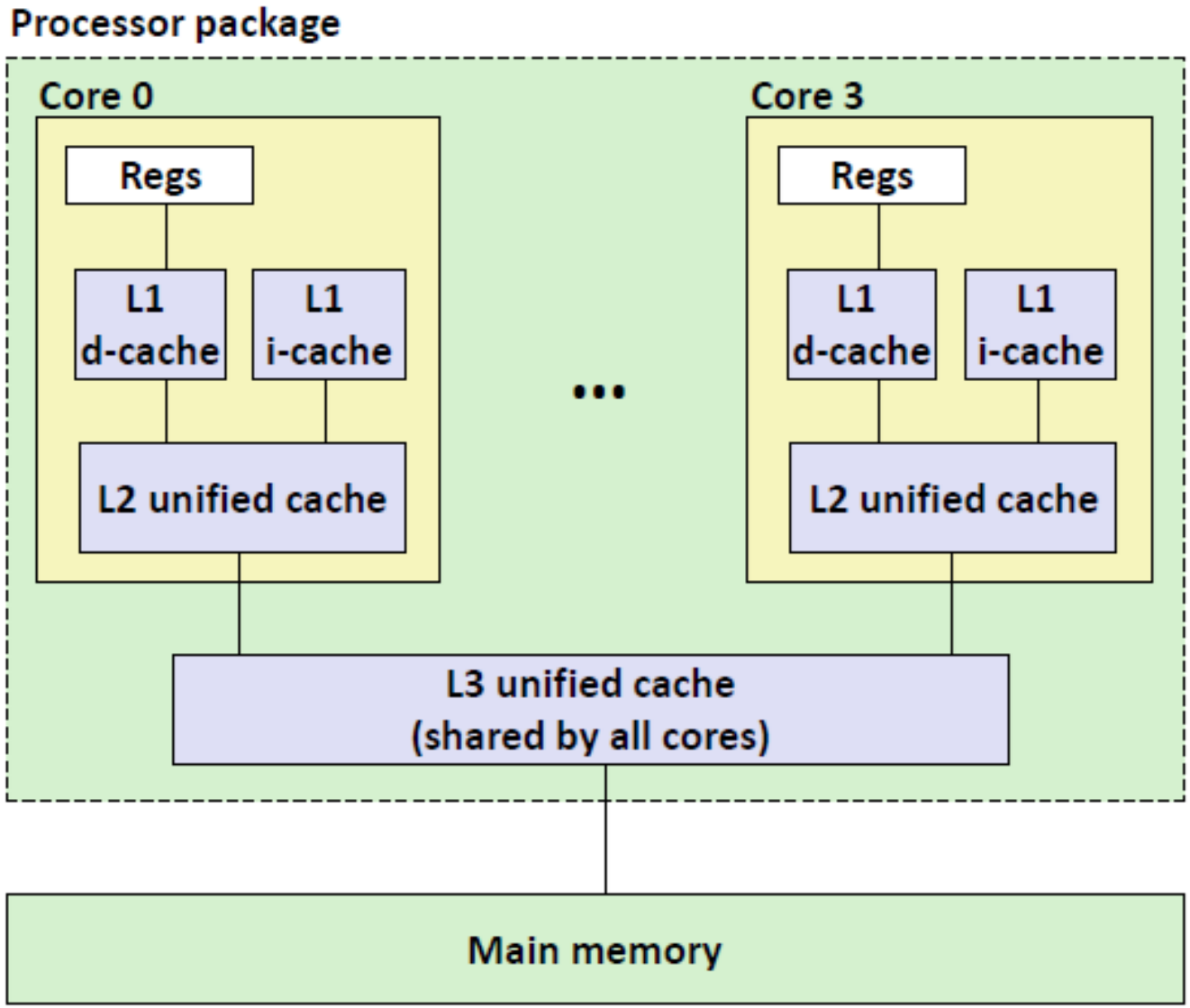
جایگاه درس مدار منطقی و معماری کامپیوتر



مفهوم

Cache

Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 10 cycles

L3 unified cache:
8 MB, 16-way,
Access: 40-75 cycles

Block size:
64 bytes for all caches

زبان برنامه‌نویسی سطح بالا

کامپایلر

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

زبان برنامه‌نویسی اسمبلی

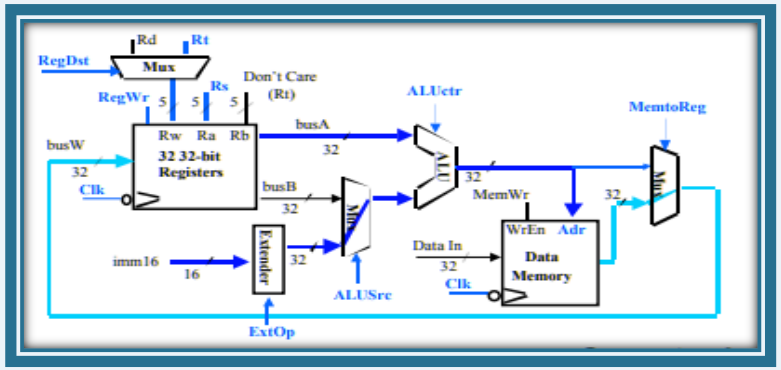
اسمبلر

```
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
```

زبان ماشین

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

مفسر ماشین



مشخصه‌ی سیگنال کنترل

مقدمه
الگوهای برنامه‌نویسی
انواع دسته‌بندی زبان‌های برنامه‌نویسی
انواع حوزه‌های کاری برنامه‌نویسی
مقایسه‌ی زبان‌های برنامه‌نویسی مختلف

Cache **مفهوم**

- در کلان‌داده (Big Data) مفهوم کش کردن (Caching) به معنای ذخیره‌سازی موقت داده‌های پرکاربرد یا پرتکرار در حافظه سریع‌تر مثل RAM تا در درخواست‌های بعدی، سیستم مجبور نباشد دوباره آن داده‌ها را از منبع اصلی (مثل دیسک یا پایگاه‌داده توزیع‌شده) بخواند. این کار سرعت پردازش، پاسخ‌گویی، و کارایی کل سامانه را به‌طور چشمگیری افزایش می‌دهد.

مثال کاربردی در کلان‌داده

- وقتی در Spark داده‌ای را می‌خوانی و چند مرحله محاسبه روی آن انجام می‌دهی، اگر همان داده را چند بار در مراحل بعد نیاز داشته باشی، می‌توانی آن را **cache** کنی تا Spark مجبور نباشد دوباره از فایل اصلی در HDFS یا S3 بخواند.
- مثلاً در PySpark:

Copy code python

```python
df = spark.read.parquet("hdfs://data/sales/")
df.cache()  # داده در حافظه نگه داشته می‌شود
df.count()  # اولین بار داده بارگذاری و cache می‌شود
```

مزایای Caching **در** Big Data

- ⚡ **افزایش سرعت پردازش** (کاهش زمان اجرای job ها)
- 💾 **کاهش بار روی سیستم ذخیره‌سازی** (مثل HDFS یا S3)
- 🔁 **تسریع در محاسبات تکراری یا iterative algorithms** (مثلاً در یادگیری ماشین)
- 🧠 **استفاده بهینه از حافظه توزیع‌شده بین نودها**

🔹 انواع Cache در معماری کلان‌داده:

نوع Cache	محل قرارگیری	مثال
In-memory cache	حافظه RAM گره‌ها	Spark Cache, Redis
Distributed cache	بین چند سرور	Hadoop DistributedCache, Ignite
Query cache	در سطح پایگاه داده	Hive, Presto, BigQuery
Result cache	نتایج پرس‌وجوی پرتکرار	Druid, ElasticSearch
Application cache	در لایه برنامه یا API	Redis, Memcached

تفاوت Cache با Storage:

ویژگی	Cache	Storage
هدف	سرعت	ماندگاری
دوام داده	موقتی	دائم
محل	RAM یا SSD	دیسک یا HDFS
سرعت دسترسی	بسیار بالا	پایین‌تر

۱. Caching

در

Apache Hadoop

۱. Caching در Apache Hadoop

- **نقش:**
- Hadoop به‌صورت سنتی روی **دیسک** (**HDFS**) کار می‌کند، نه حافظه.
بنابراین caching در Hadoop به معنی **ذخیره‌ی موقت داده‌های پرکاربرد در حافظه‌ی نودها** (**DataNodes**) است تا از خواندن مکرر از دیسک جلوگیری شود.
- 🔹 **ابزار و روش‌ها:**
- از نسخه‌های جدید Hadoop، قابلیتی به نام **DistributedCache** یا **Centralized Cache Management** اضافه شده است.
- داده‌هایی که در چند Job استفاده می‌شوند (مثلاً lookup tables) را می‌توان در حافظه نودها cacheکرد.
- 🔹 **هدف:**
- کاهش I/Oدیسک در MapReduce و افزایش سرعت Jobهای تکراری.

۰۲ Caching

در

Apache Spark

۲. Caching در Apache Spark

- **نقش:**
- Sparkبرای **محاسبات در حافظه (In-Memory Computing)** طراحی شده است، پس caching یکی از اجزای اصلی آن است.
- 🔹 **نحوه استفاده:**
- در Spark، هر **RDD**، **DataFrame** یا **Dataset** را می‌توان cache یا persist کرد:

```python
rdd.cache()          # در حافظه ذخیره می‌شود
rdd.persist(StorageLevel.MEMORY_AND_DISK)  # در حافظه یا در صورت پر بودن، روی دیسک
```

- **کاربردها:**
- در الگوریتم‌های یادگیری ماشین که داده‌ها بارها تکرار می‌شوند (iterative).
- در تحلیل‌های تعاملی (Interactive Analytics).
- در pipelineهای ETL با چند مرحله وابسته به یک منبع داده واحد.
- 🔹 **مزایا:**
- چندین برابر سریع‌تر از Hadoop MapReduce.
- Sparkخود به‌صورت خودکار حافظه نودها را بین jobها تقسیم می‌کند.

۳. Caching
در
Apache Kafka

۳. Caching در Apache Kafka

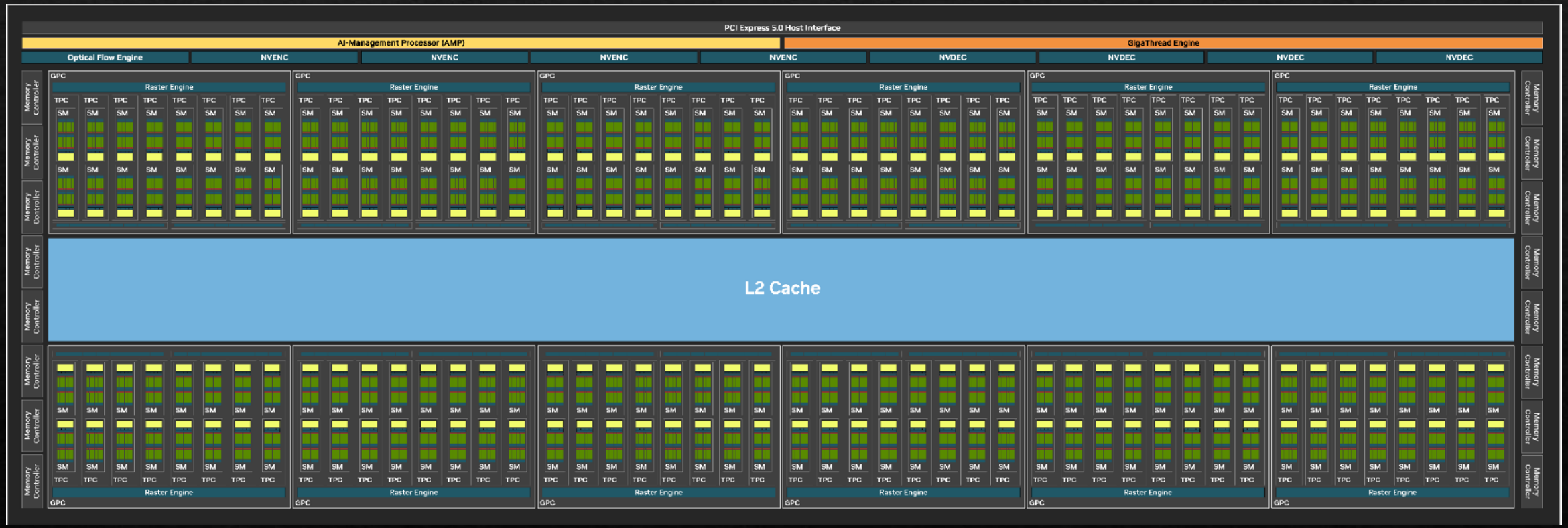
- 🔹 **نقش:**
- Kafka برخلاف Hadoop و Spark، **سیستم پردازش یا تحلیل مستقیم داده نیست** بلکه **پیام‌رسان توزیع‌شده** (**streaming platform**) است.
- 🔹 **محل استفاده از cache:**
- Kafka از caching در چند لایه استفاده می‌کند:
- **Page Cache سیستم‌عامل:**
داده‌ها قبل از نوشتن روی دیسک یا هنگام خواندن از آن در حافظه kernel نگه داشته می‌شوند.
- **Broker Cache:**
برای کاهش latency در ارسال پیام‌ها به consumerها.
- **Streams API Cache:**
در Kafka Streams، نتایج stateful processing (مثل join یا aggregation) در حافظه cache می‌شوند تا نیاز به نوشتن مداوم در state storeنباشد.
- 🔹 **هدف:**
- کاهش تأخیر در پردازش جریان داده‌ها
- جلوگیری از overloadروی دیسک
- پایداری و کارایی بالا در throughput

مقایسه‌ی نهایی

ویژگی	Hadoop	Spark	Kafka
نوع سیستم	Batch Processing	In-Memory & Batch/Streaming	Streaming & Messaging
هدف از Cache	کاهش I/O دیسک	تسریع محاسبات تکراری	کاهش latency پیام‌ها
محل Cache	DataNode Memory	Executor Memory	OS Page Cache و Stream Cache
کنترل توسط کاربر	محدود	کامل (cache/persist)	تا حدی (configurable)
دوام داده	تا پایان Job	تا زمانی که job زنده است	تا زمان retention message

تعریف
SM
در
GPU

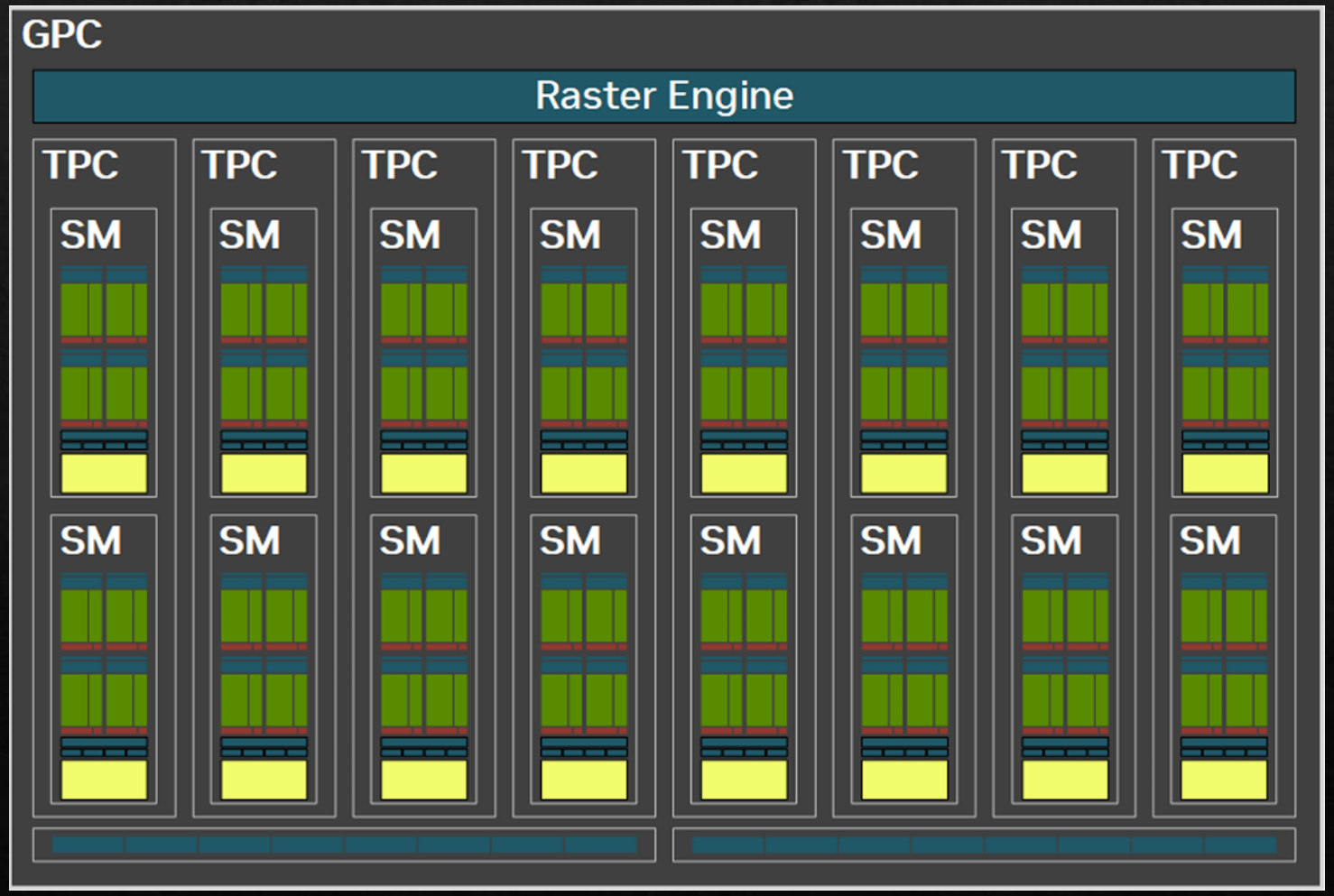
GB202 Die Shot



GB202 Components

- GPCs
- Memory Controllers.
- Cache.
- AMP & GigaThread Engine.
- NVENC / NVDEC.
- Optical Flow Engine.
- PCI Express 5.0 Host Interface.

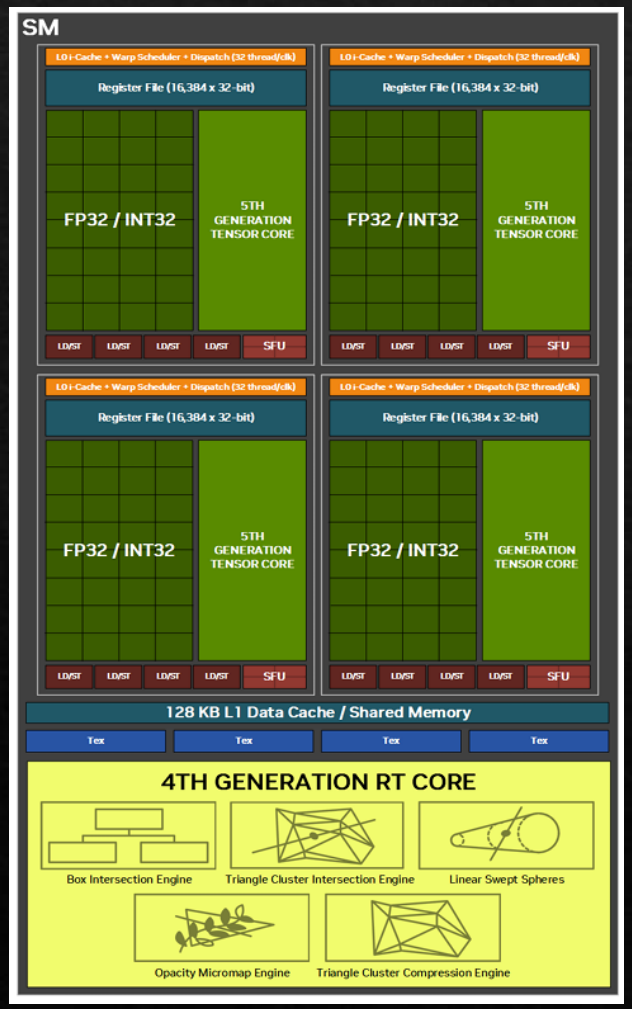
GPC

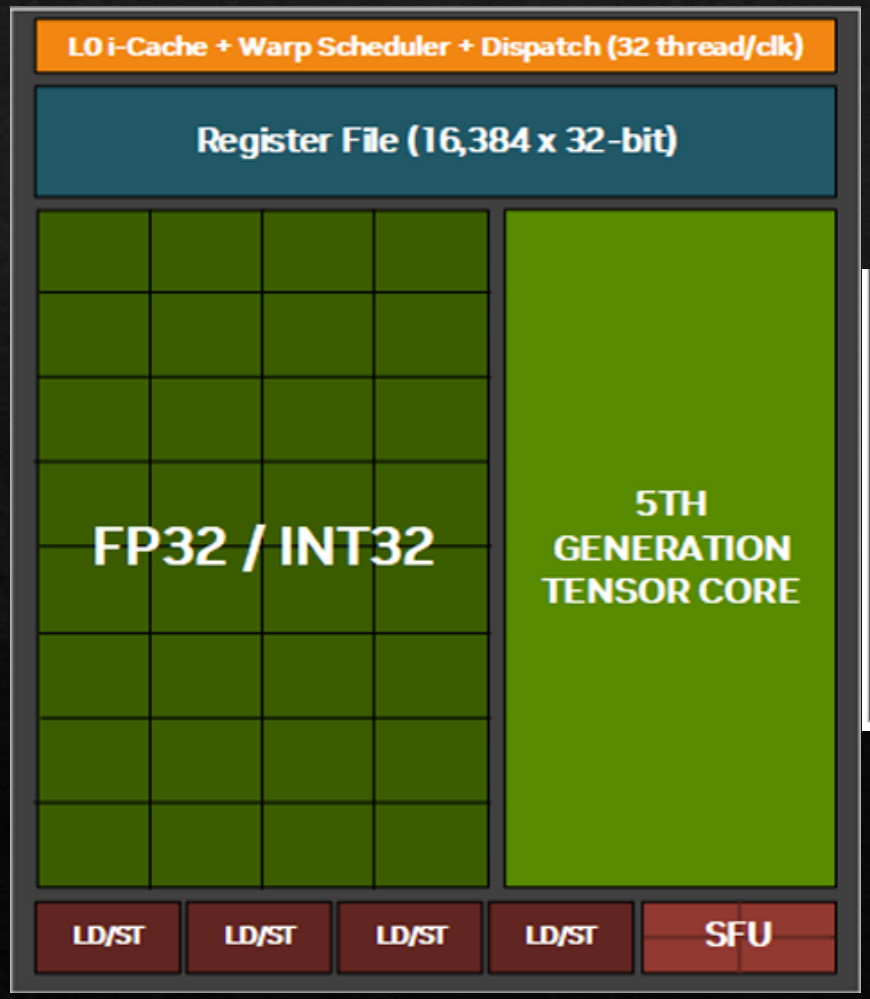

GPC
Raster Engine
TPC
TPC
TPC
TPC
TPC
TPC
TPC
TPC
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM
SM

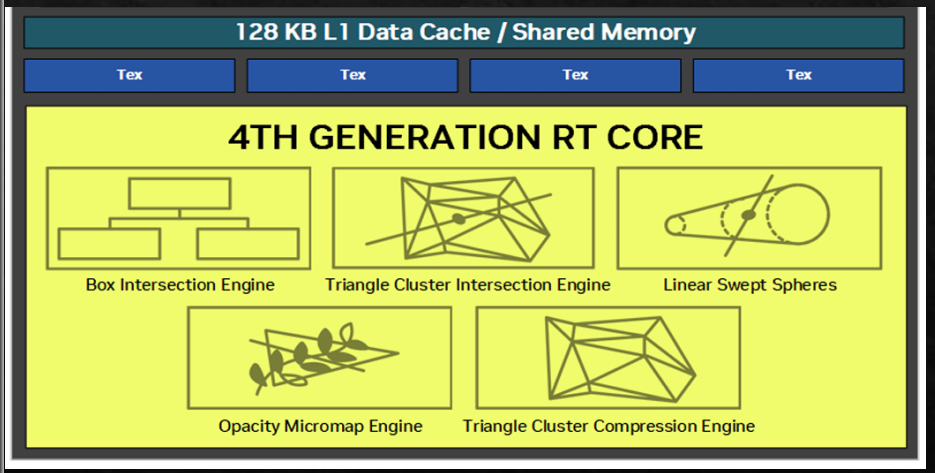
GPC cont.

- Fundamental building block of the GPU's architecture.
- Stands for **G**raphics **P**rocessing **C**lusters.
- The GB202 has 12 GPCs.
- Contains:
  - Raster Engines.
  - TCPs.
  - ROPs.
- Provides scalable graphics and compute performance.

SM


SM
L0 i-Cache + Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
FP32 / INT32
5TH GENERATION TENSOR CORE
LD/ST
SFU
128 KB L1 Data Cache / Shared Memory
Tex
4TH GENERATION RT CORE
Box Intersection Engine
Triangle Cluster Intersection Engine
Linear Swept Spheres
Opacity Micromap Engine
Triangle Cluster Compression Engine


L0 i-Cache + Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
FP32 / INT32
5TH GENERATION TENSOR CORE
LD/ST
LD/ST
LD/ST
LD/ST
SFU


128 KB L1 Data Cache / Shared Memory
Tex
Tex
Tex
Tex
4TH GENERATION RT CORE
Box Intersection Engine
Triangle Cluster Intersection Engine
Linear Swept Spheres
Opacity Micromap Engine
Triangle Cluster Compression Engine

تعریف SM

- **SM (Streaming Multiprocessor)** در واقع **واحد پردازش اصلی GPU** است.
هر GPU از ده‌ها تا صدها SMتشکیل شده، و هر SM خودش شامل **ده‌ها هسته (CUDA Cores)**، **حافظه‌ها**، و **واحدهای محاسباتی اختصاصی** است.
- به زبان ساده:
- هر SM را می‌توان مثل "یک مینی‌پردازنده‌ی چندهسته‌ای" درون GPU تصور کرد.

ساختار یک SM (در معماری‌های مدرن مثل H200)

هر SM شامل اجزای زیر است:

مؤلفه	توضیح
CUDA Cores	هسته‌های محاسباتی برای عملیات عددی (مثل جمع، ضرب، ماتریس).
Tensor Cores	واحدهای مخصوص برای شتاب‌دهی به عملیات ماتریسی (در یادگیری عمیق).
Shared Memory / L1 Cache	حافظه‌ی سریع و محلی که بین threadهای همان SM مشترک است.
Register File	سریع‌ترین حافظه، مخصوص هر thread.
Warp Scheduler	زمان‌بند اجرای threadها (واحد کنترل).
Load/Store Units	واحدهای دسترسی به حافظه (برای خواندن و نوشتن داده).
Special Function Units (SFU)	برای محاسبات خاص (مثل سینوس، ریشه دوم، لگاریتم).

SM

- The fundamental execution unit of NVIDIA GPUs.
- Includes:
  - CUDA Cores(FP32/INT32 ALUs): handle general math & logic.
  - Tensor Cores: accelerate AI/ML workloads
  - RT Core: for ray/triangle intersection.
  - Warp Schedulers & Dispatch Units: decide which warps (groups of 32 threads) run each cycle.
  - Texture Units: perform texture fetches and filtering.
  - Load/Store Units (LD/ST): handle memory access (global/local memory reads/writes).
  - Register File: private storage per thread.
  - Shared Memory / L1 Cache: memory accessible by all threads in the SM.

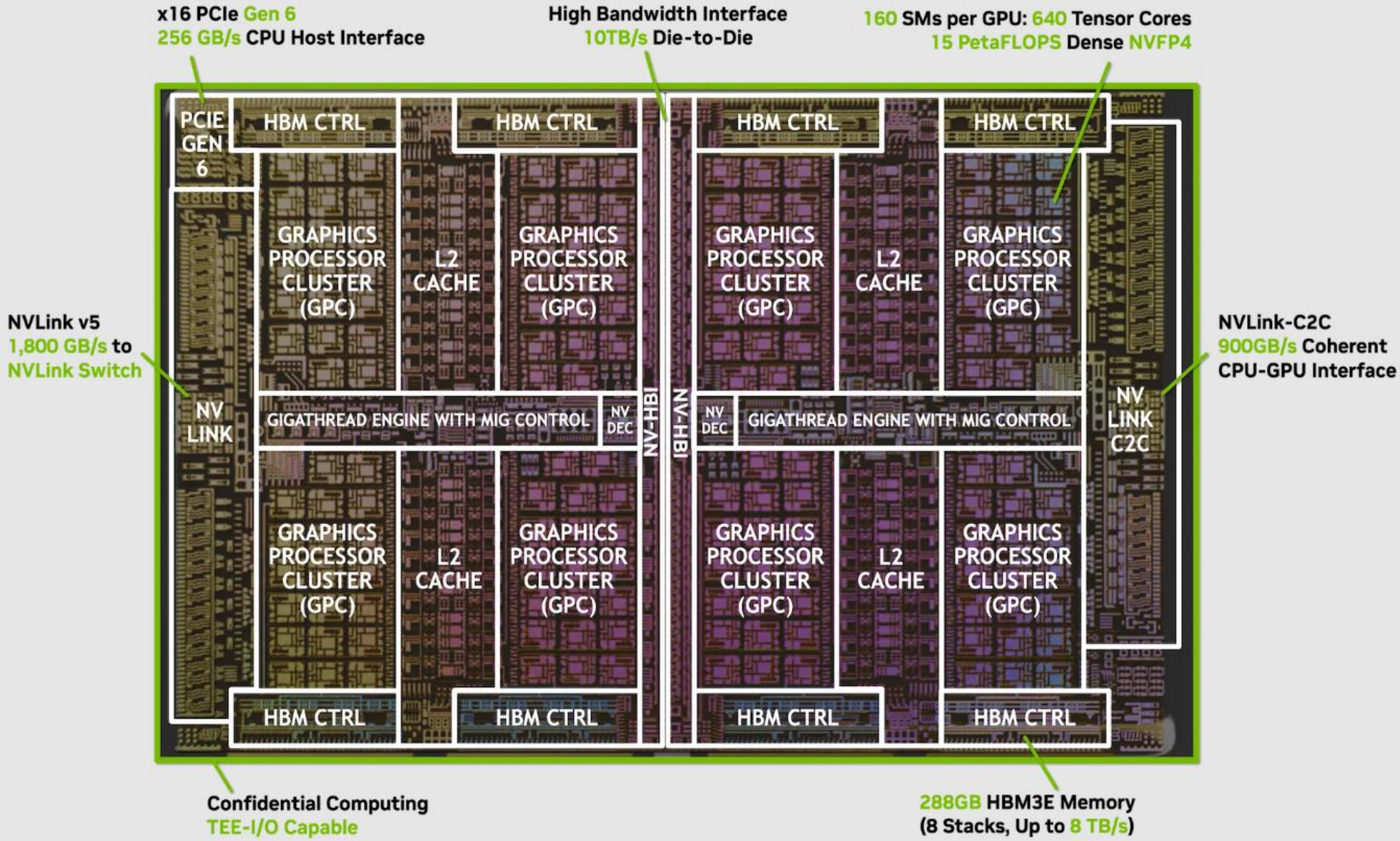
SM Cont.

- Example of SMIT(Single Instruction Multiple Threads)

```
for (i=0; i<32; i++)
    C[i] = A[i] + B[i];
```

- One ADD instruction is issued.
- All 32 threads perform the addition simultaneously, each with different A[i], B[i].
- 64 warps per each SM(4 can be active in each cylce).
- 4 warp schedulers in each SM.

NVIDIA Blackwell Ultra GPU



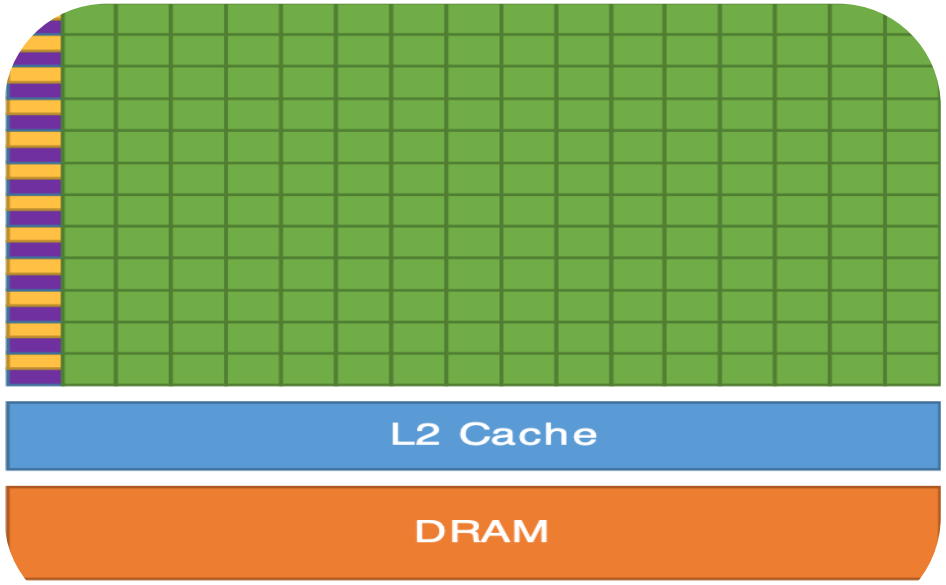
بلاک های معماری بلک ول

GPC
(Graphics Processing Clusters)

- Raster Engine
- ROP (Raster Operations) partitions
- Texture/Processing Clusters (TPCs)
  - PolyMorph (geometry) engine
  - Streaming multiprocessors (SMs)
    - Texture units
    - 256 KB register file and 128 KB of configurable L1/shared memory
    - CUDA ALUs
    - Fifth-generation Tensor Cores
    - Fourth-generation RT Core

انواع حافظه در GPU

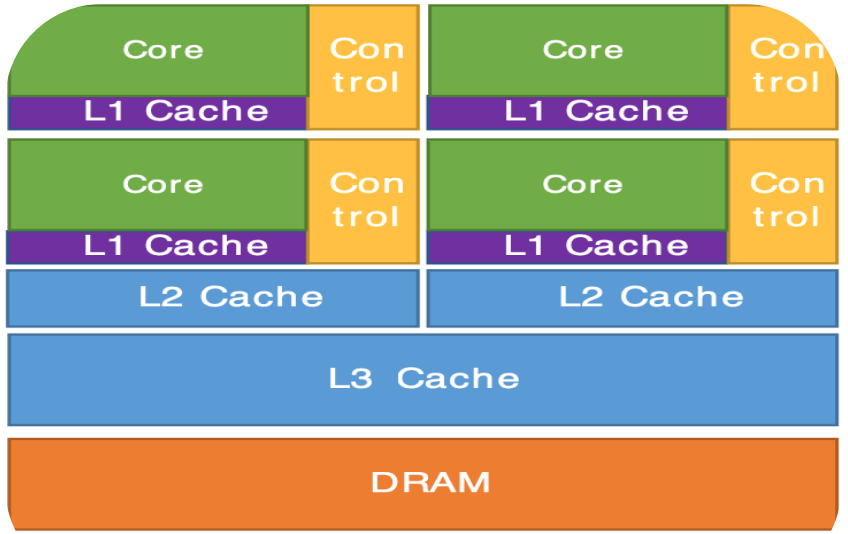
معماری پردازنده‌ها (CPU و GPU)



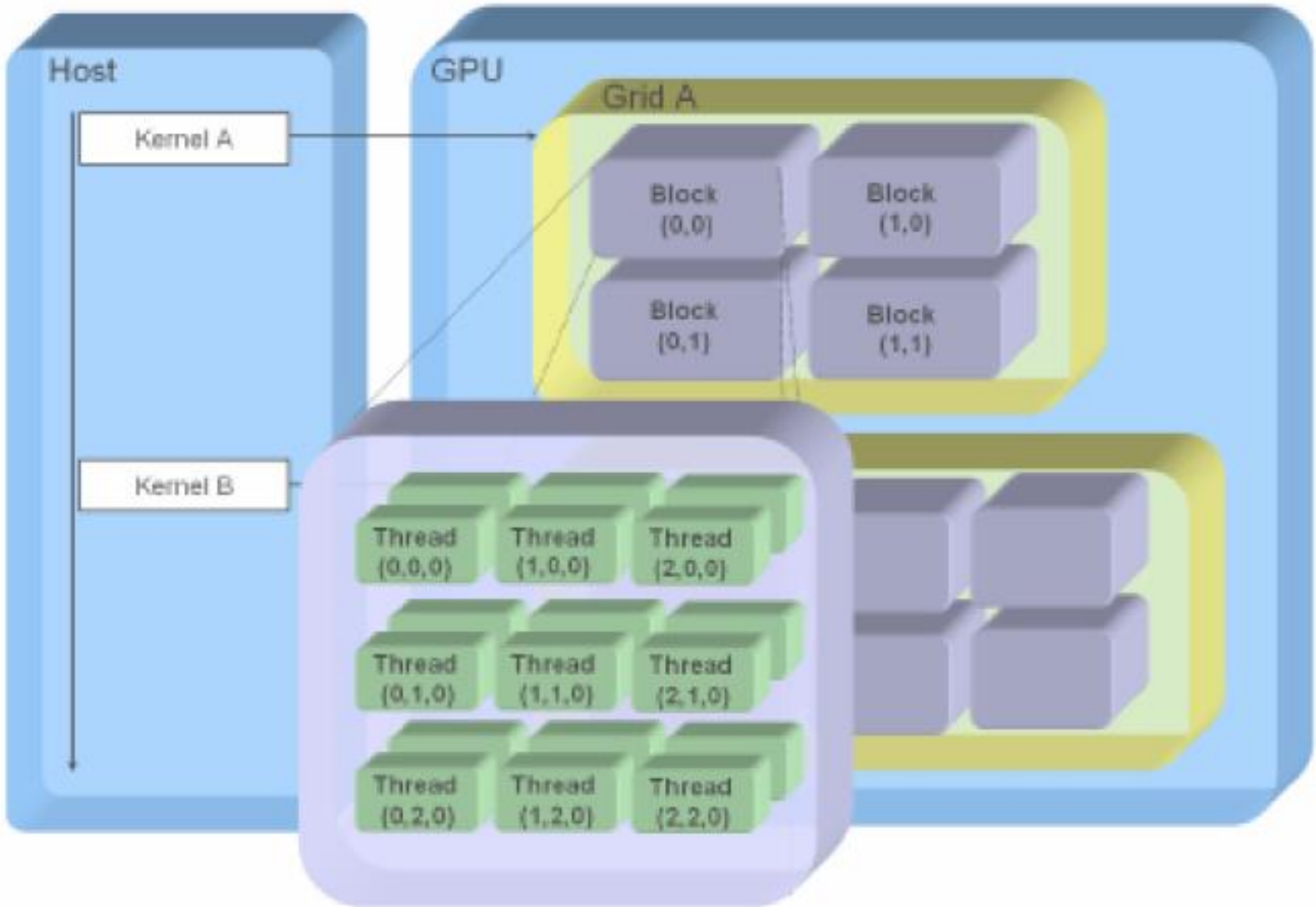
GPU

- دارای هزاران هسته
- طراحی شده برای پردازش موازی
- مناسب برای workloadهایی با داده‌های مستقل (مثل خوانش‌های NGS)
- بهره‌برداری از حافظه shared برای کاهش latency

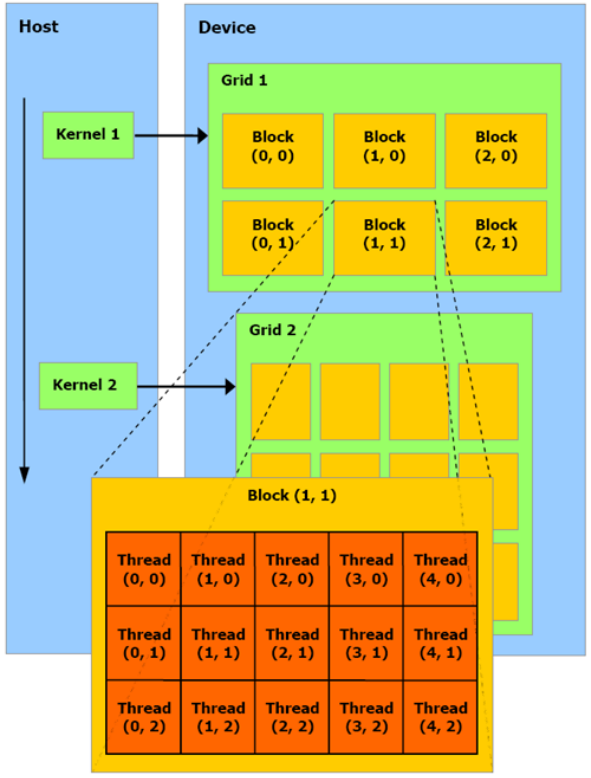
- مناسب برای پردازش های ترتیبی
- دارای تعداد محدودی هسته (4-64 هسته)
- کش‌های بزرگ و بهینه‌سازی‌شده برای latency پایین

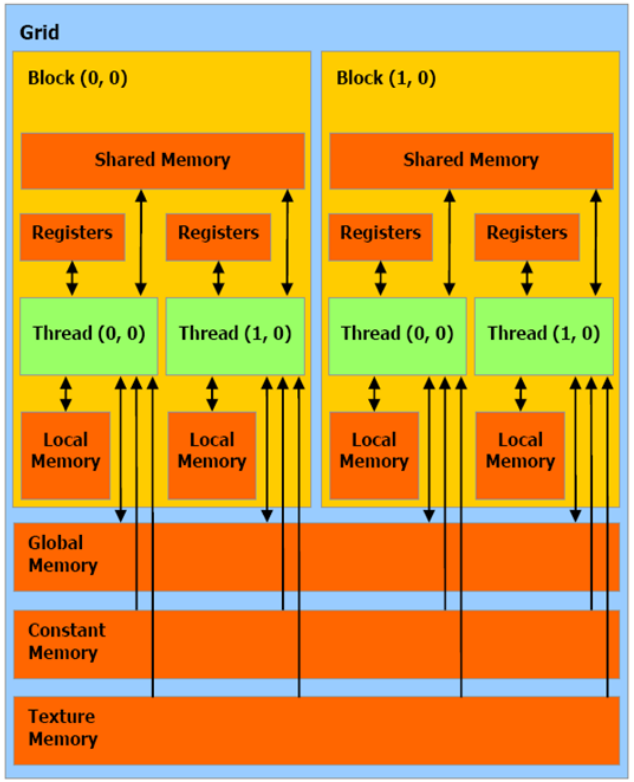


CPU


Host
GPU
Grid A
Kernel A
Kernel B
Block (0,0)
Block (1,0)
Block (0,1)
Block (1,1)
Thread (0,0,0)
Thread (1,0,0)
Thread (2,0,0)
Thread (0,1,0)
Thread (1,1,0)
Thread (2,1,0)
Thread (0,2,0)
Thread (1,2,0)
Thread (2,2,0)

اجزای اصلی GPU





۱ Grid

۲ Block

۳ Thread

۴ Memory

۱. Global Memory (حافظه‌ی سراسری)

- 🔹 **تعریف:**
- Global Memory در واقع **حافظه‌ی اصلی GPU** است (اغلب از نوع HBM3 یا HBM3e در H200).
همه‌ی **Threads**(رشته‌ها) در همه‌ی **Blocks** به آن دسترسی دارند.
- 🔹 **ویژگی‌ها:**
- **ظرفیت زیاد:** چند ده گیگابایت (مثلاً ۱۴۱ GBدر H200).
- **سرعت نسبتاً پایین‌تر:** دسترسی به آن صدها چرخه‌ی کلاک طول می‌کشد.
- **دسترسی جهانی:** همه‌ی threadها از همه‌ی blocksمی‌توانند به آن بنویسند یا بخوانند.
- **پایدار:** داده‌ها تا پایان اجرای کرنل در دسترس می‌مانند.
- 🔹 **معایب:**
- تاخیر بالا (High Latency)
- پهنای باند محدود نسبت به حافظه‌های نزدیک‌تر (مثل L1 cacheیا shared memory)

۲. Shared Memory (حافظه‌ی اشتراکی)

- **تعریف:**
- Shared Memory نوعی **حافظه‌ی کوچک‌تر اما بسیار سریع‌تر** است که فقط میان **threadهای یک block مشخص** مشترک است.
در واقع می‌توان آن را **حافظه‌ی سطح L1 برای GPU threads** دانست.
- 🔹 **ویژگی‌ها:**
- **ظرفیت محدود:** معمولاً ۱۰۰ تا ۲۵۶ کیلوبایت در هر SM (Streaming Multiprocessor).
- **سرعت بسیار بالا:** تأخیر چند چرخه‌ی کلاک (خیلی سریع‌تر از global memory).
- **اشتراک در سطح block:** فقط threadهای همان block می‌توانند به آن دسترسی داشته باشند.
- **برنامه‌پذیر:** توسط برنامه‌نویس CUDA به‌صورت صریح تعریف و استفاده می‌شود.
- 🔹 **مثال در CUDA:**

Copy code cpp

```cpp
__global__ void example(int *input, int *output) {
    __shared__ int cache[256];  // تعریف حافظه اشتراکی در هر block

    int tid = threadIdx.x;
    cache[tid] = input[tid];    // داده را از حافظه سراسری می‌خوانیم
    __syncthreads();            // همگام‌سازی threadها

    output[tid] = cache[tid] * 2; // محاسبه با داده‌ی local
}
```

⚡ تفاوت‌های کلیدی

ویژگی	Global Memory	Shared Memory
محدوده‌ی دسترسی	همه‌ی threadها در همه‌ی blocks	فقط threadهای همان block
ظرفیت	بسیار زیاد (GBها)	بسیار کم (KBها)
سرعت	کند (صدها cycle)	سریع (چند cycle)
کنترل توسط برنامه‌نویس	غیرمستقیم	مستقیم (با `__shared__`)
محل فیزیکی	در خارج از SM (در HBM)	درون SM
دوام داده	در طول اجرای کرنل	تا پایان block
کاربرد	ذخیره‌ی داده‌های اصلی یا ورودی‌های بزرگ	بافر موقت برای محاسبات محلی و تکراری

نتیجه نهایی

- محاسبات خیلی زمانبر با تعداد تکرار زیاد اگر در Shared memory بجای Global memory قابل اجراست، بهتر است با تغییر کد در آنجا انجام شود تا سرعت به طرز قابل توجهی افزایش پیدا کند.
- حافظه Shared به نوعی نقش Cache memory را در GPU ایفا می کند و بین Thread های یک بلوک مشترک است.
- پایپ لاین ها هم باید به نحوی طراحی شوند که هر گروه از داده فقط داخل یک Shared memory هدایت شود.

مثالی از کاربرد کلان داده
در پردازش داده ژنتیک

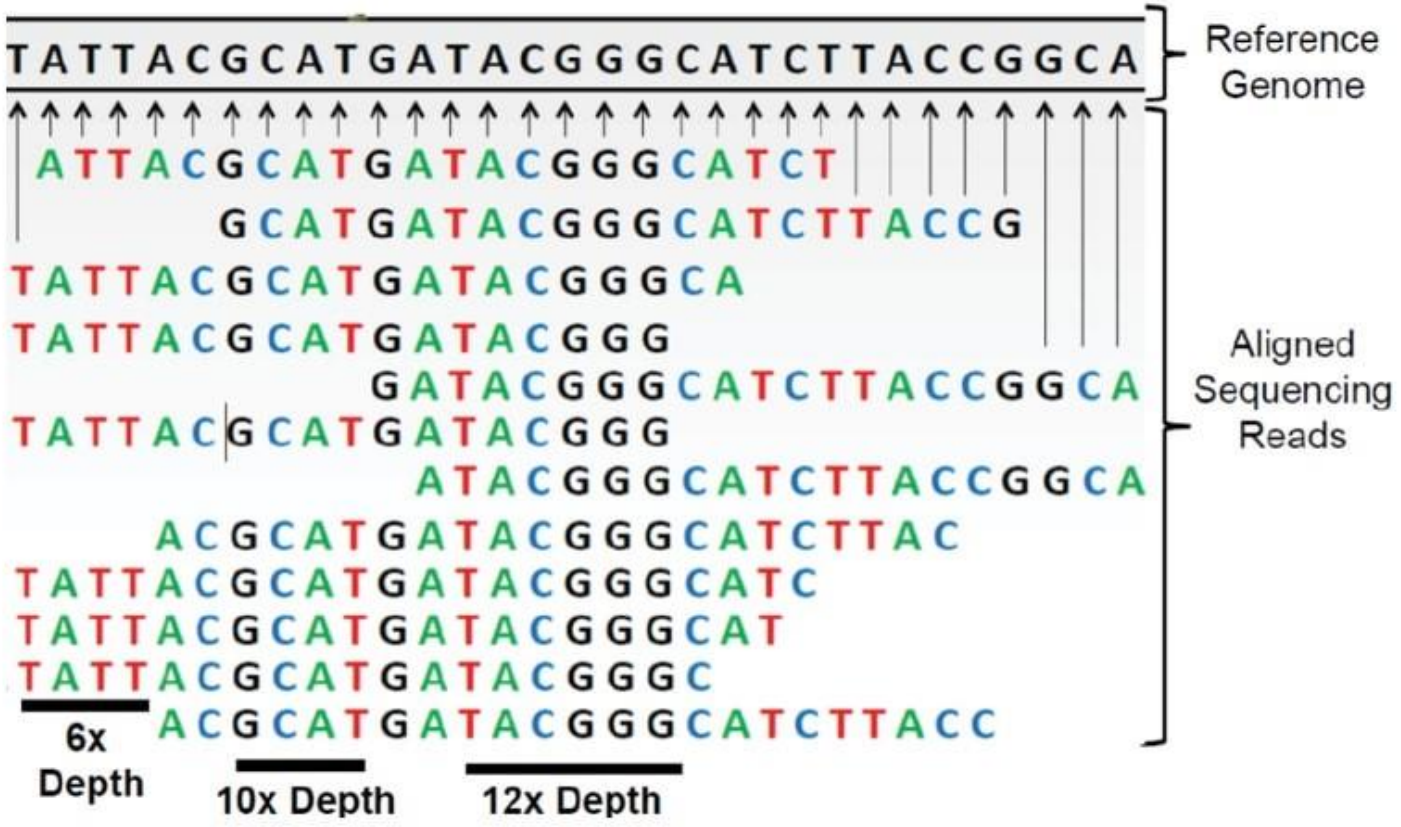
Template DNA

Short reads from Sequencer (FASTQ)

Mapped reads (SAM/BAM-file)

Sequence Alignment

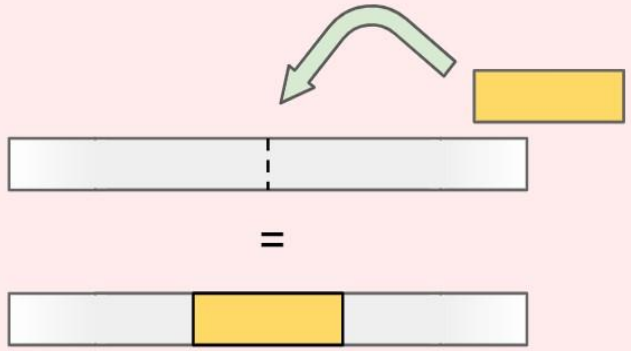
- All reads are aligned to a reference genome like GRCh38 and hg19 in this step.
- Two alignment files **SAM** (Sequence Alignment Map) and **BAM** (Binary Alignment Map) are produced.



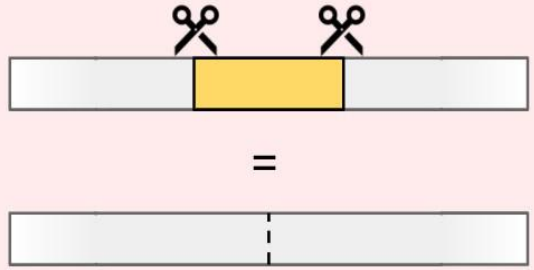
SV Types

Destructive (non-balanced)

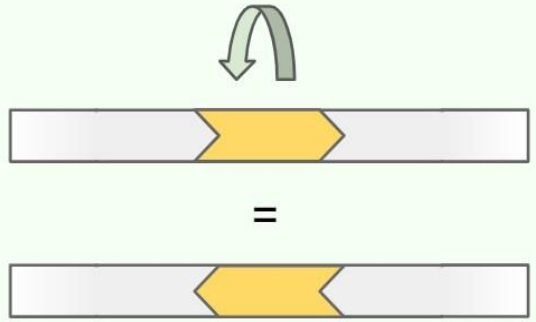
Non-destructive (balanced)



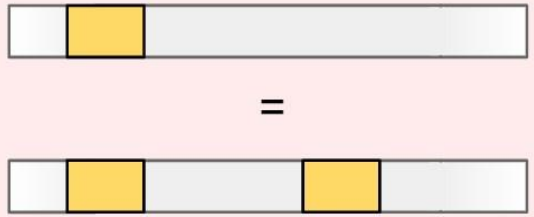
Insertion



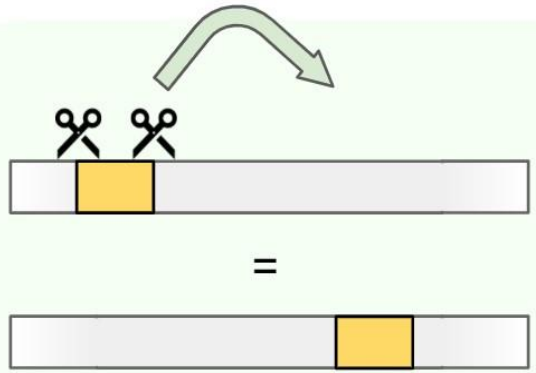
Deletion



Inversion



Interspersed Duplication



Translocation
(inter- or intra- chromosomal)

Tandem Duplication