TensorCore – vysvětlení pojmu

TensorCore (česky tensorové jádro) je specializovaná výpočetní jednotka, která je součástí moderních grafických procesorů (GPU) od společnosti NVIDIA. Je navržena pro akceleraci výpočtů s tenzory – vícerozměrnými maticemi – a především pro výkonné provádění operací typu matrix-multiply-accumulate (MMA), což jsou základní stavební kameny deep-learningových algoritmů.

Historie

```
  Volta (2017) – první generace GPU s TensorCores (např. NVIDIA V100).
  Turing (2018) – rozšíření o podporu inferencí a ray-tracing
(RTX 2000-serie).
  Ampere (2020) – zvýšený počet a vyšší přesnost (FP16, BF16, TF32,
INT8/INT4).
  Ada Lovelace (2022) – další optimalizace a podpora nových datových typů.
```

Architektura a princip fungování

```
  MMA jednotka – provádí operaci C = A × B + C na 4 × 4 blokových maticích
najednou.
  Datové typy – podporuje různé formáty:
      FP16 (half-precision) – původní podpora.
      BF16 (brain-floating-point) – pro vyšší dynamický rozsah.
      TF32 – kombinace FP32 a FP16, optimalizovaná pro trénink.
      INT8 / INT4 – určené pro inferenci s nízkou přesností.
  Paralelismus – každé TensorCore může paralelně zpracovávat až 64 bitových
operací, což vede k třicetinásobnému zrychlení oproti klasickým FP32
výpočtům na stejném GPU.
```

Využití

```
  Deep Learning – trénink a inference neuronových sítí (CNN, RNN,
Transformer).
  High-Performance Computing (HPC) – simulace, lineární algebra, vědecké
výpočty.
  AI akcelerátory – integrace do serverových a edge zařízení (např. NVIDIA
Jetson).
```

Výhody TensorCores

```
  Vyšší propustnost – až několik TFLOPS (tera-floating-point operations per
second) pro FP16/TF32.
  Úspora energie – více výpočtů na watt ve srovnání s tradičními CUDA jádry.
  Jednoduchá integrace – využívá se pomocí knihoven jako cuBLAS, cuDNN,
TensorRT a frameworků (PyTorch, TensorFlow) bez nutnosti psát nízko-úrovňový
```

```
kód.
```

=== Příklad kódu (CUDA) ===

```
 #include <cuda_fp16.h> #include <stdio.h>

global void matMulTensorCore(half *A, half *B, float *C, int N) { // Využití
WMMA API (CUDA 10+) wmma::fragment<wmma::matrix_a, 16, 16, 16, half,
wmma::row_major> a_frag; wmma::fragment<wmma::matrix_b, 16, 16, 16, half,
wmma::col_major> b_frag; wmma::fragment<wmma::accumulator, 16, 16, 16,
float> c_frag;

// Načtení sub-matice do fragmentů
wmma::load_matrix_sync(a_frag, A, N);
wmma::load_matrix_sync(b_frag, B, N);
wmma::fill_fragment(c_frag, 0.0f);

// Výpočet C = A × B + C
wmma::mma_sync(c_frag, a_frag, b_frag, c_frag);

// Uložení výsledku
wmma::store_matrix_sync(C, c_frag, N, wmma::mem_row_major);

}
```

Další zdroje

```
  [[https://developer.nvidia.com/tensor-cores|NVIDIA – Tensor Cores –
oficiální dokumentace]]
  [[https://arxiv.org/abs/1803.08968|“Mixed Precision Training” –
Micikevicius et al., 2018]]
  [[https://pytorch.org/docs/stable/generated/torch.cuda.amp.html|PyTorch –
Automatic Mixed Precision (AMP)]]
```

From:
http://serviceit.cz/ - **IT ENCYKLOPEDIE**

Permanent link:
http://serviceit.cz/doku.php?id=tensorcore

Last update: **2026/01/02 21:11**