GPU alkalmazás részecskefizikai szimulációkban

Nagy Máté Ferenc
MTA KFKI RMKI – ALICE csoport
ELTE TTK – Alkalmazott fizika
Témavezető:
Dr. Barnaföldi Gergely Gábor
MTA KFKI RMKI – ALICE csoport

Vázlat

1. ALICE és a GRID
2. Szimulációk, AliROOT
3. MC generátorok
4. OpenCL és Cypress vs. Fermi
5. TRandom4, eredmények

ALICE és a GRID

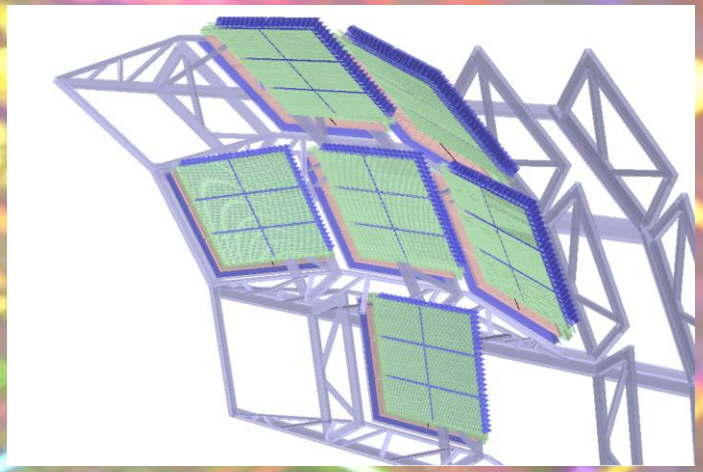
- ALICE egyike a 3 CERN LHC óriáskísérletnek.
- A detektorokból érkező lementett adatok feldolgozásához szuperszámítógép parkra volna szükség.
- Olcsóbb és kezelhetőbb a probléma elosztott rendszerrel.
- A kísérlet tagjainak kötelező hozzájárulni a GRID rendszerhez.

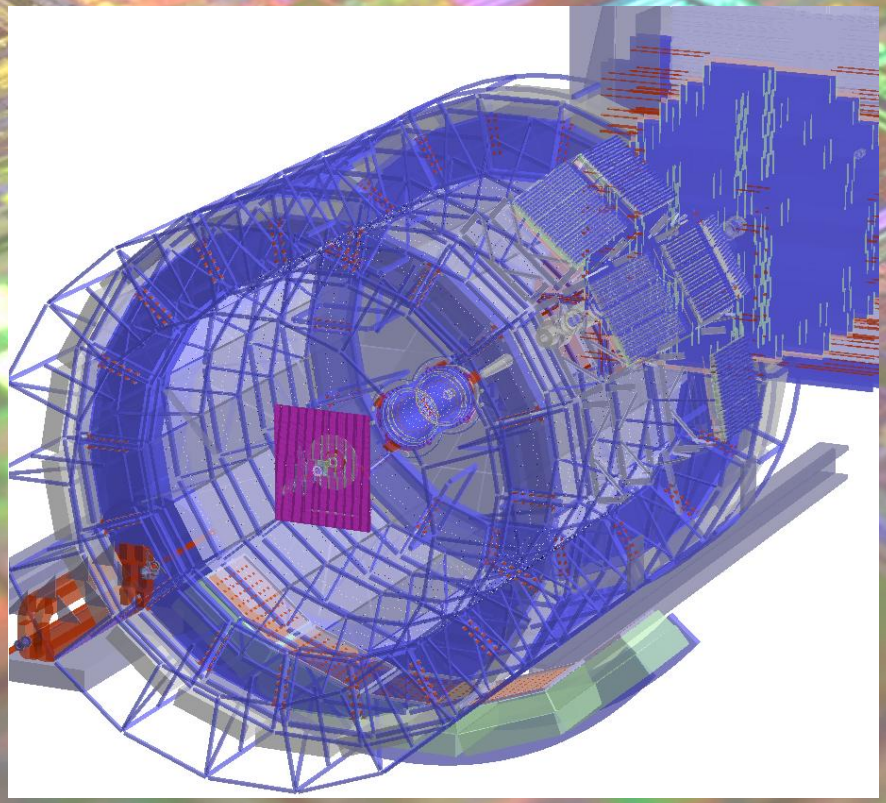
Szimulációk & AliROOT

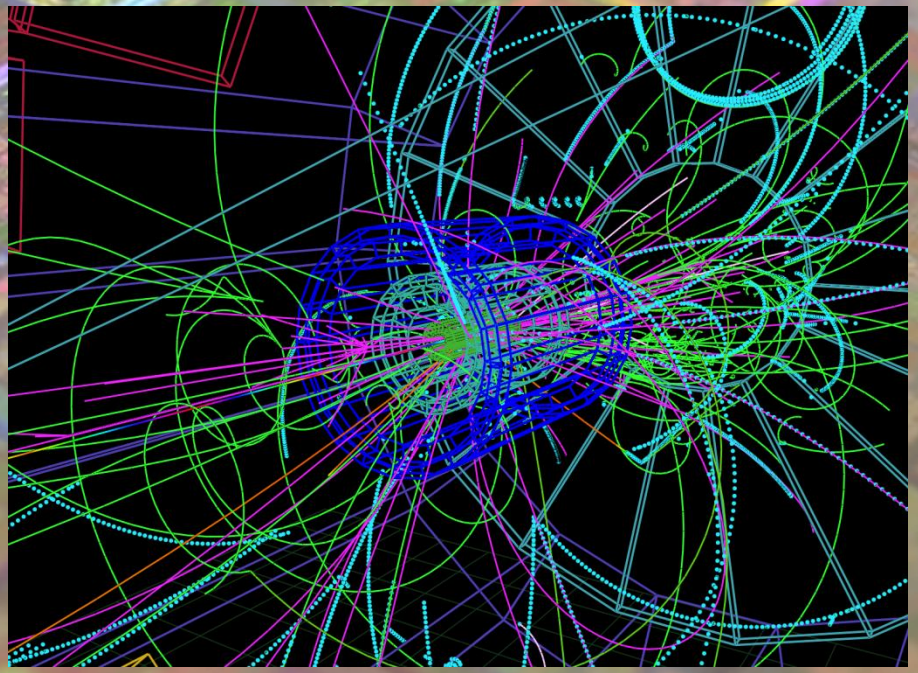
- A használt szimulációs környezet az AliROOT.
- Az elméletek következményeit szimuláljuk.
- A szimulált eredményeket összevetjük a valós mérési eredményekkel.
- Ha egyezik örülünk, ha nem egyezik tovább berhelünk.
- Cél: gyorsítás.

AliROOT

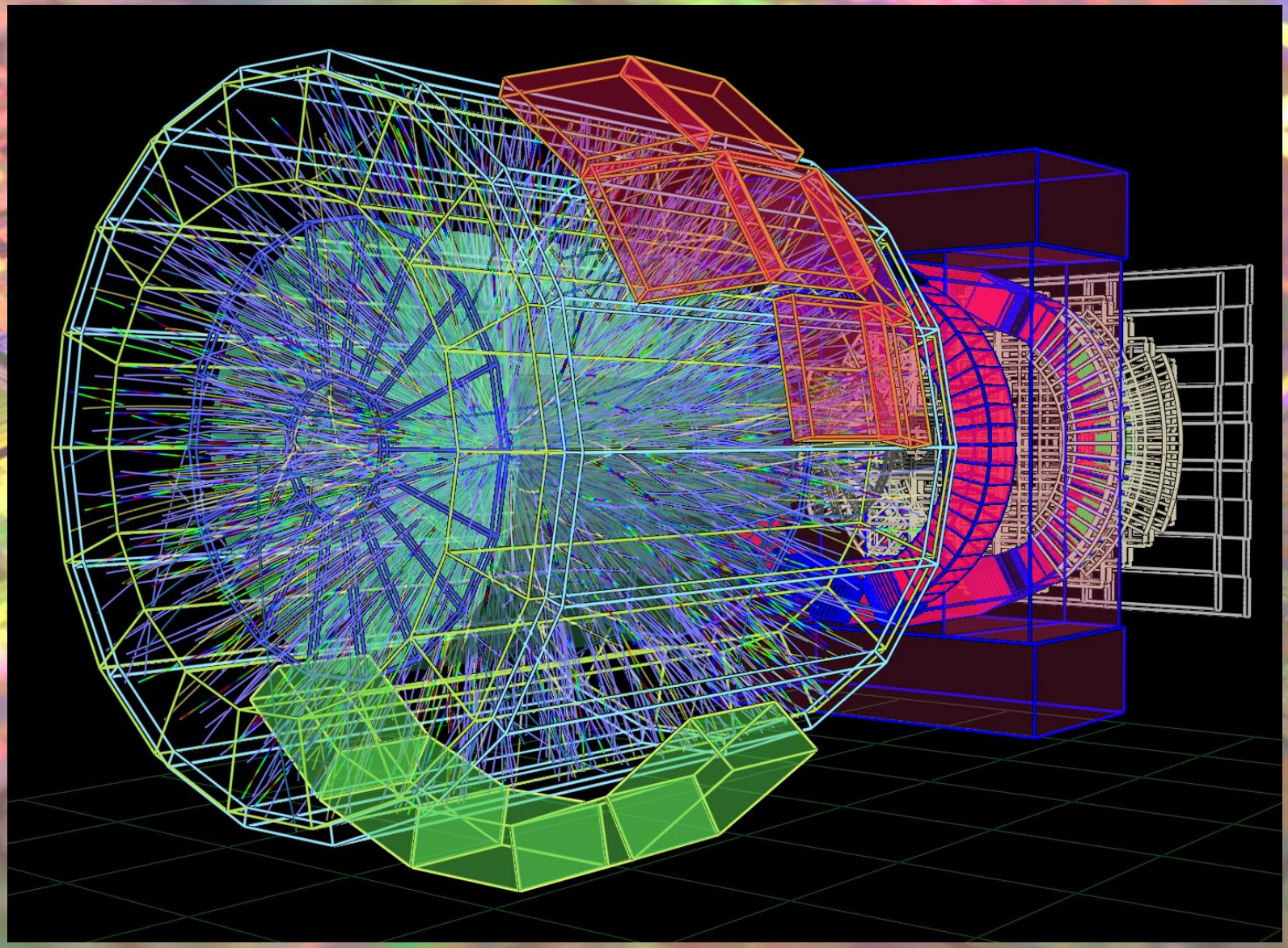
- 2,7GB-os keretrendszer, több eseménygenerátort magábafoglaló.
- Mindenkori verziója tartalmazza az aktuális ALICE detektor terveit.







Motiváció



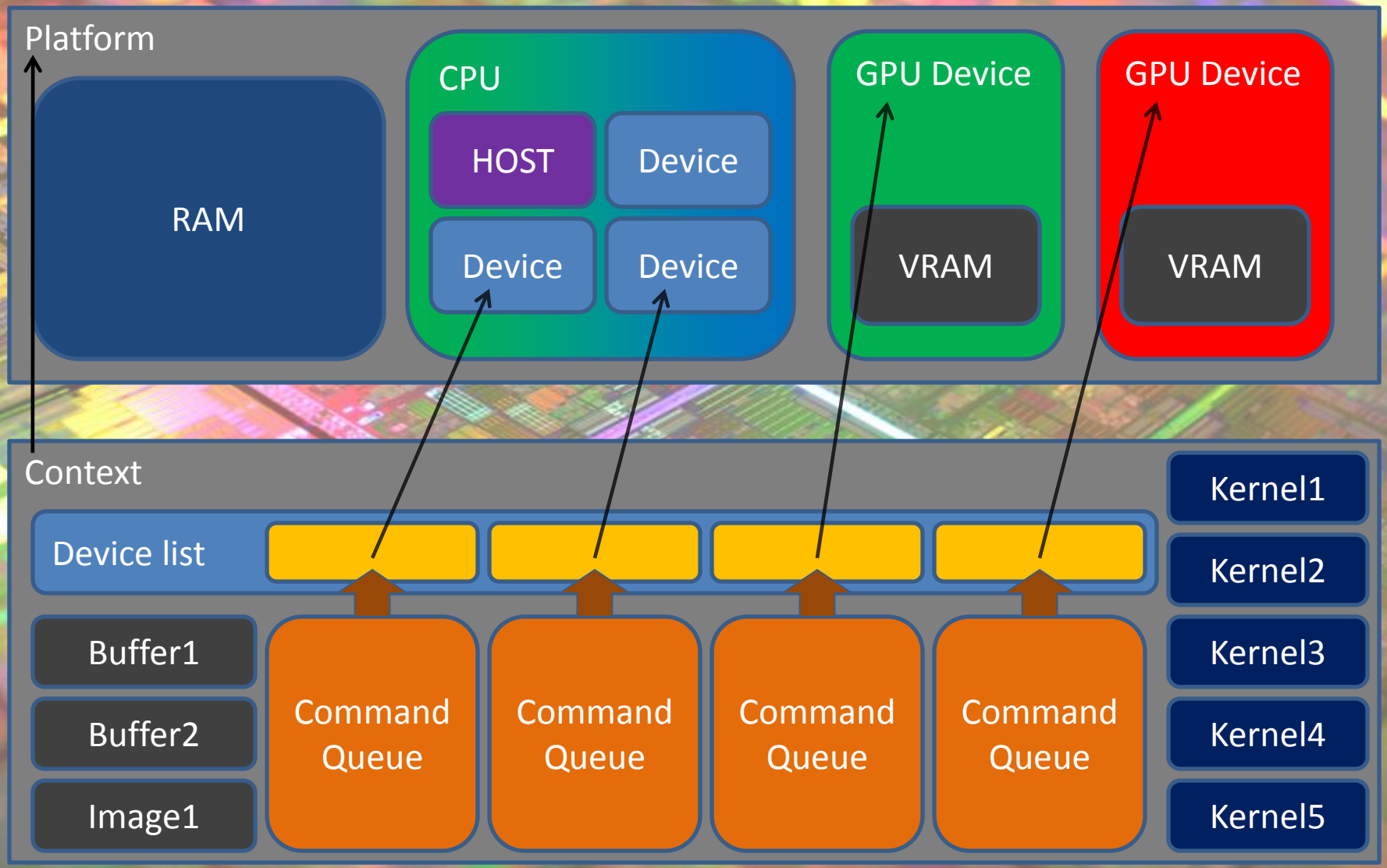
Monte-Carlo generátorok

- Monte-Carlo szimulációnak nevezünk minden olyan eljárást, amelyben egymástól függetlenül generált véletlen számok határozzák meg a rendszer fejlődését, kimenetét.
- Az ilyen szimulációk eseményeit előállító modult hívjuk MC generátornak.
- Pszeudo random véletlen számok előnyösek a szimuláció reprodukálása miatt.
- A nagy léptékű mintavételezés miatt a gyorsaság centrális fontosságú.

OpenCL

- Nyílt szabványú, gyártó- és platformfüggetlen programozási nyelv.
- GPU-n futó kód írható OpenCL szabvány C nyelven, vagy fordítható C/C++ nyelvről natív kóddá.
- Fordítók egyaránt léteznek CPU-ra és GPU-ra. StreamSDK (ATi) és CudaSDK (NVIDIA)
- Széleskörű támogatás párhuzamosítható programok írásához.
- Léteznek magasabb szintű kiterjesztések (PyOpenCL és társai)

OpenCL szerkezet

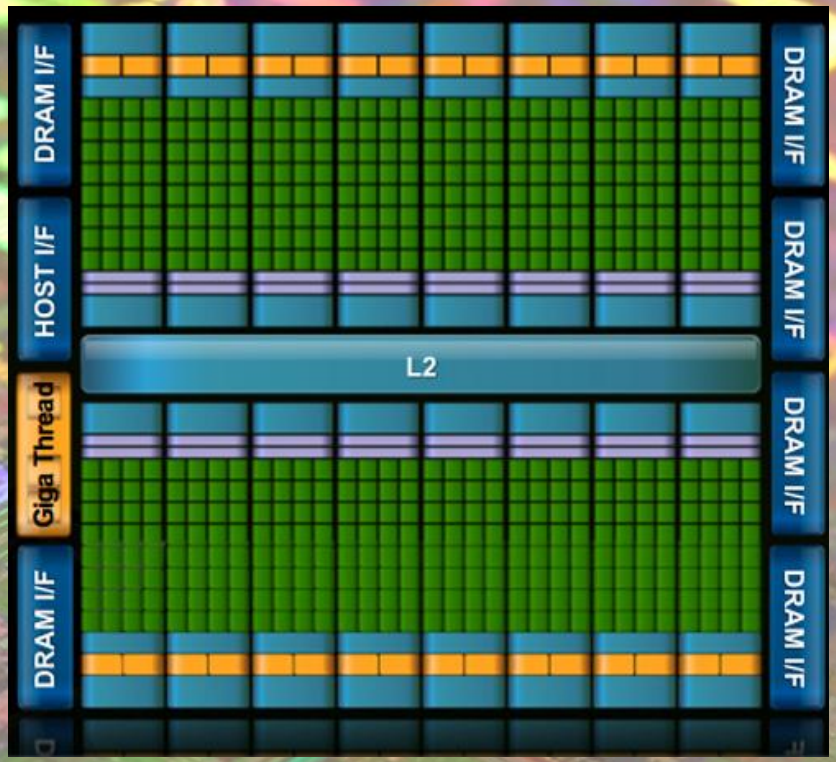


Cypress vs. Fermi

- 1600 shader processzor
- 20*80 (80 = 64 SP + 16 DP) Minden ötödik shader egység double precision
- 2 GB VRAM / GPU
- Óriási shader kapacitás, alacsony fogyasztás

- 512 shader processzor
- Minden második shader double precision
- 3-6 GB VRAM / GPU
- Óriási cache méretek, széles memória sín

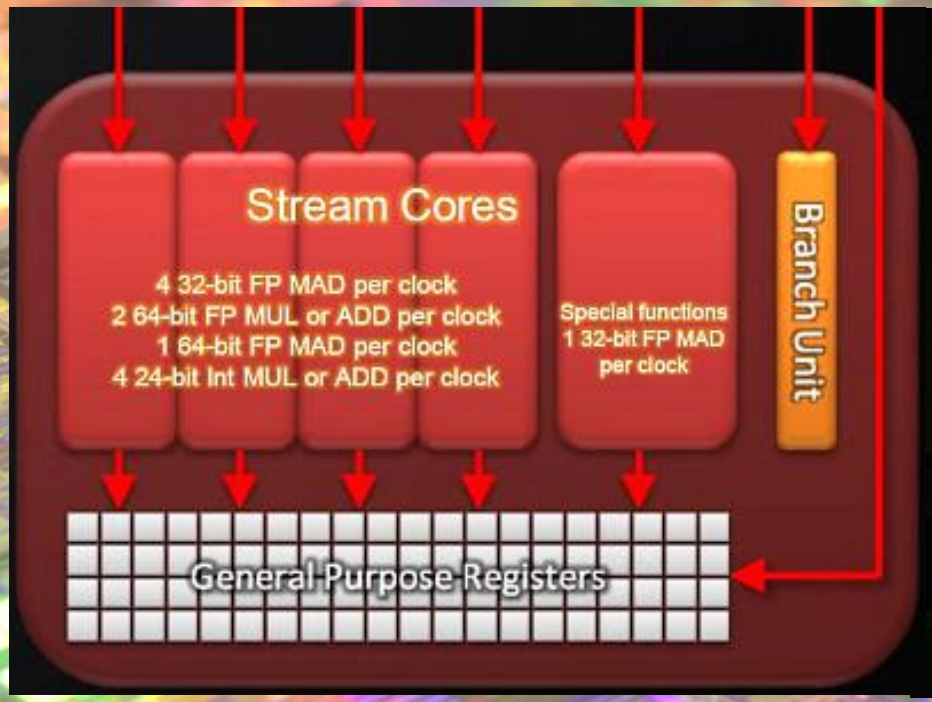
Cypress vs. Fermi

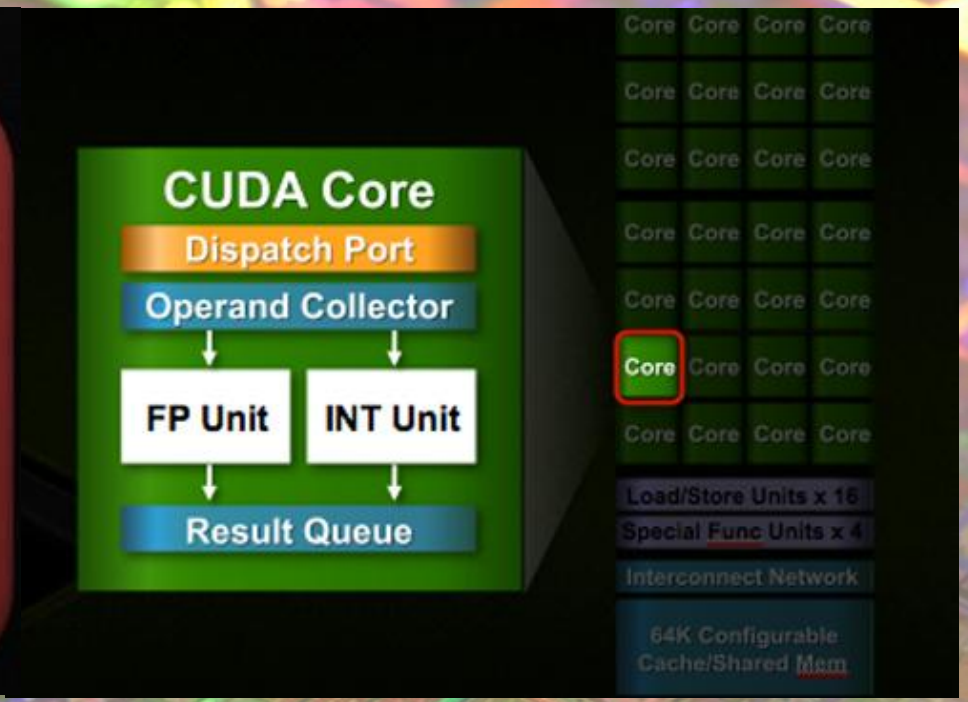


- 1600 shader processzor
- 20*80 (80 = 64 SP + 16 DP) Minden ötödik shader egység double precision
- 2 GB VRAM / GPU
- Óriási shader kapacitás, alacsony fogyasztás

- 512 shader processzor
- 16*32 (32 = 16 SP + 16 DP) Minden második shader double precision
- 3-6 GB VRAM / GPU
- Óriási cache méretek, széles memória sín

Cypress vs. Fermi





- 1 shader processzor = 1 Stream Core
- Egyszerű vektorprocesszor
- 2700 GFLOPS SP, 500 GFLOPS DP

- 1 shader processzor = 1 CUDA Core
- Külön FP és INT egység
- 1600 GFLOPS SP, 650 GFLOPS DP

grid249

- Intel Core-i7 2.66GHz, 12GB DDR3 1333MHz RAM, 3 db Radeon 5970

AliROOT gyorsítás

- Cél: Véletlenszám generátorhoz GPU gyorsítást írni.
- A ROOT matematikai osztálya 3 alap generátort definiál. Ezekből a Mersenne-Twister algoritmust használó TRandom3 a legelterjedtebb.
- Definiáltam egy új modult (TRandom4) amely GPU gyorsítással végzi a random szám generálást.

TRandom4

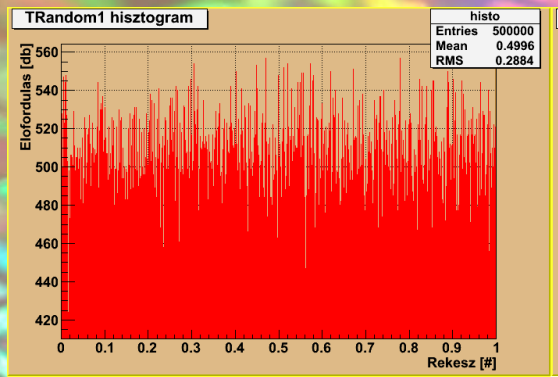
- Konstruktor tartalmaz mindent a tényleges szám generálásig:
- Platform lekérdezés
- Context létrehozás
- Device info query
- Kernel fordítás
- Command queue létrehozás
- Buffer létrehozás
- Random seedek device-ra küldése

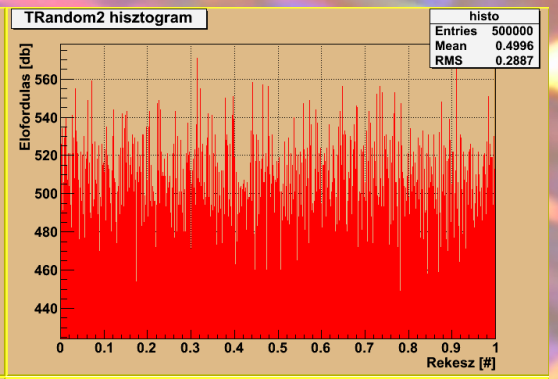
TRandom4

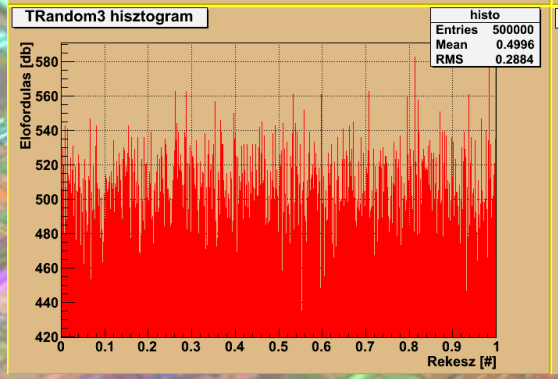
- A gyorsabb működés érdekében nem lehet egyesével számot generálni.
- Egy RAM-ban lévő buffert tölt fel véletlen számokkal. Kérésnél a bufferből másolja ki az eredményt
- Esetszétválasztás lassítja, aszerint hogy a bufferben maradt-e annyi szám, amennyit kértek.
- RAM-ból másolás plussz művelet a CPU implementációhoz képest.

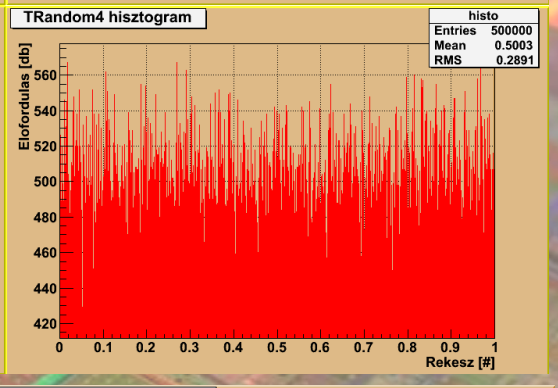
Eredmény

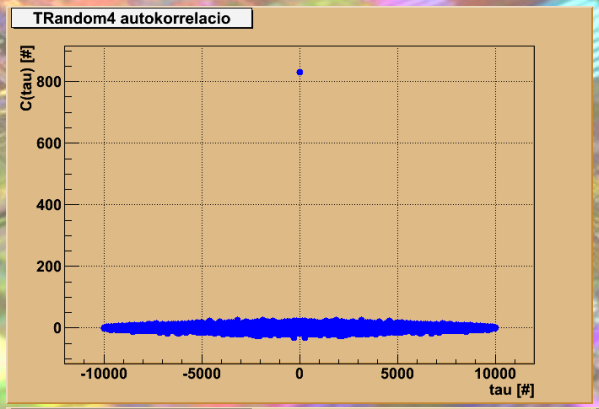
- TRandom4 kimenetének statisztikai vizsgálata

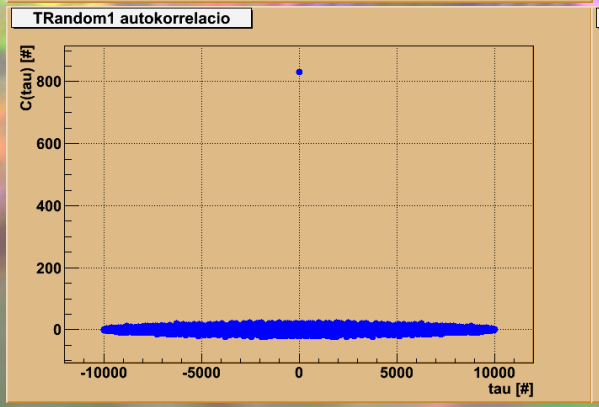


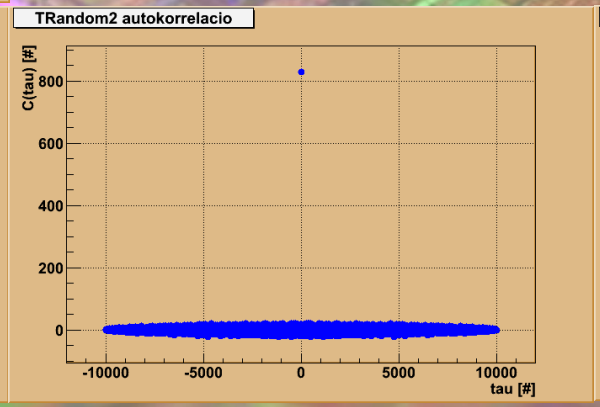


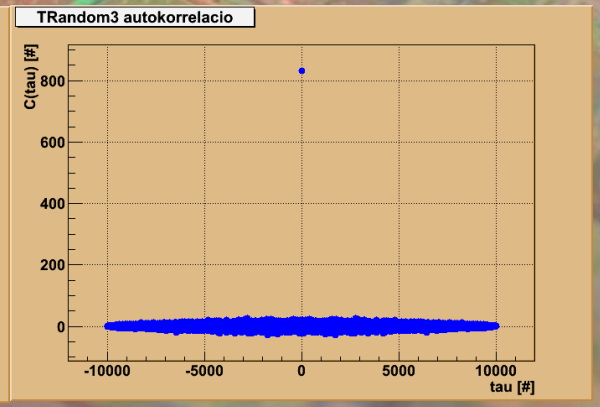












Eredmény

Mért idő	Generálási idő [ns / db]	Ráta [db / ns]
TRandom3 kernel	6,697	0,14932
TRandom4 kernel	0,411	2,43309
TRandom4 teljes	14,7	0,06802

- Teljes generálási idő bő kétszer lassabb.
- Ok a túlságosan nagy overhead a feladat méretéhez képest.
- Kernel idő kevesebb, mint tizenhatod része a CPU verziónak.

Továbbiakban

- Monte-Carlo generátor nagyobb, összefüggő részét kiültetni GPU-ra.
- Végeredményben teljes MC generátor után más jól párhuzamosítható részek kerülnek górcső alá.
- GRID rendszerbe történő tényleges integráció.