GPU-k a kisérleti és elméleti gravitációkutatásban

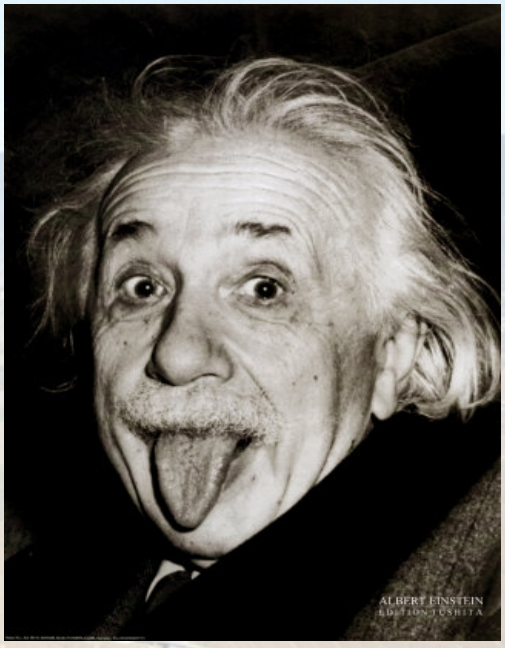
RMKI GPU nap – 2010

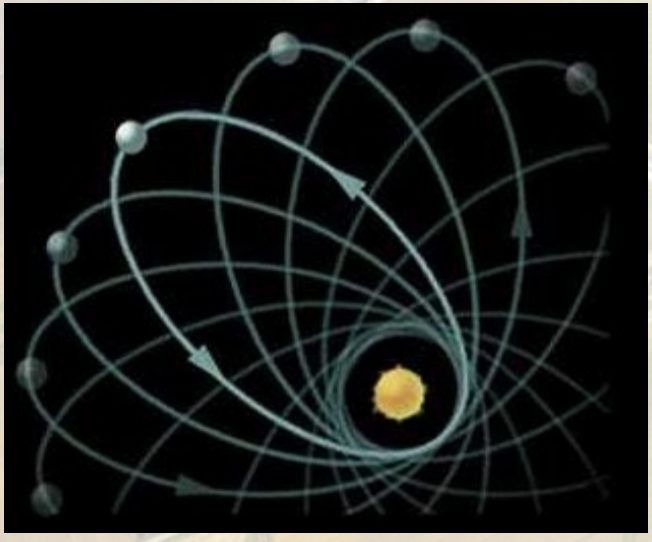
Debreczeni Gergely

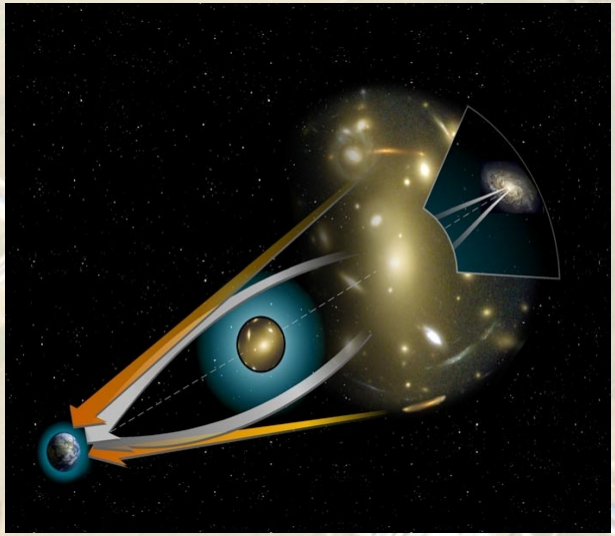
(MTA KFKI RMKI)
RMKI VIRGO Csoport

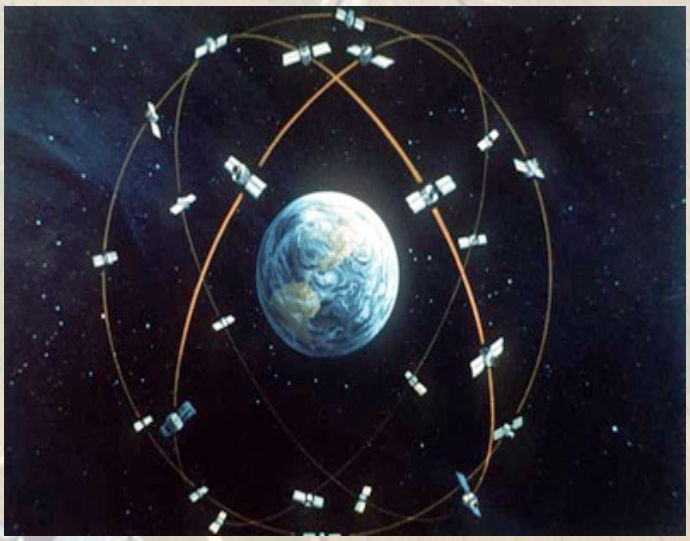
A gravitációs hullámokról

- Einstein általános relativitáselméletének számos jóslata bebizonyosodott már
- A perihélium elfordulás
- A gravitációs lencsézés
- Az idő múlásának 'sebessége' gravitációs terekben megváltozik
- De a gravitációs hullámok létezésére még csak közvetett bizonyítékunk van ...
- Közvetlen kimutatásuk kutatásunk célja !



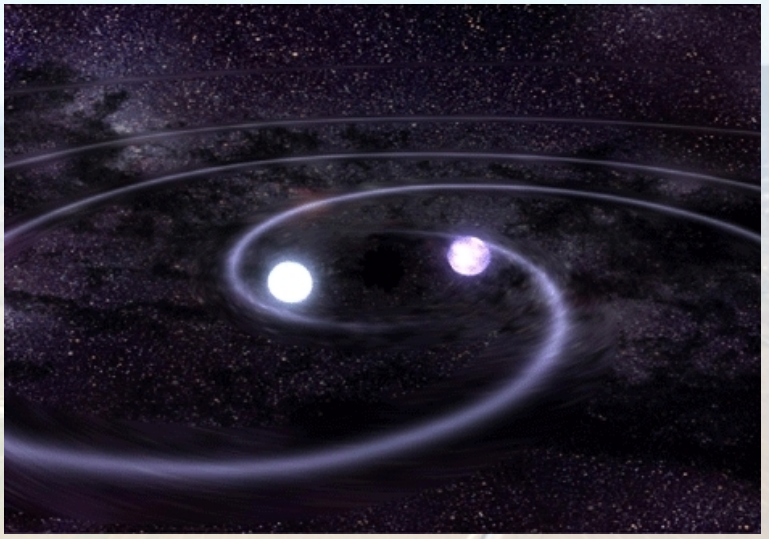


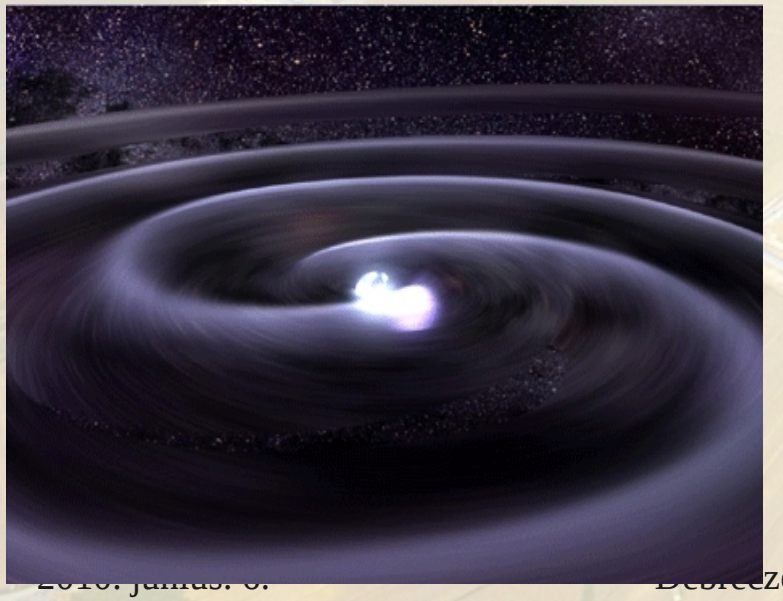


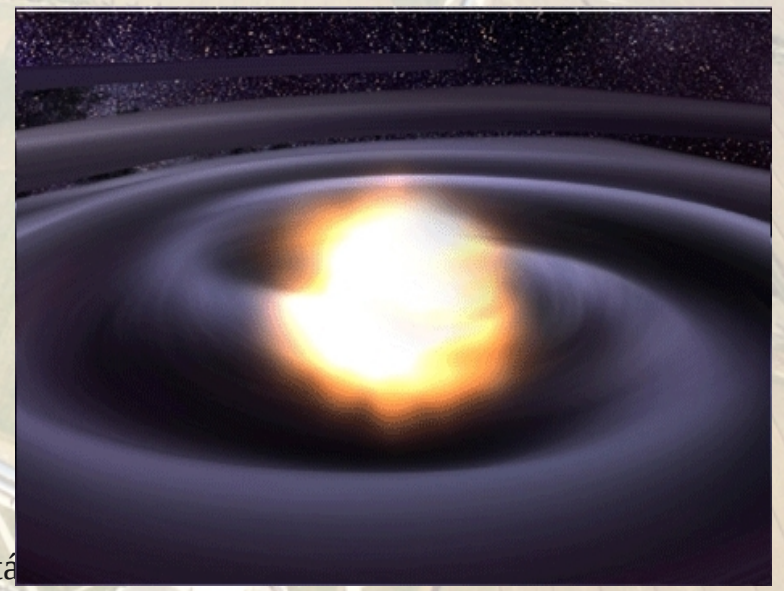


A gravitációs hullámokról

- Ha nagy tömegek mozognak elég gyorsan, akkor gravitációs hullámokat bocsátanak ki.
- A gravitációs hullámok magának a térnek rezgései és nem a térben haladó rezgések.
- A tér rezgései miatt megváltoznak a 'távolságok' a dolgok között.
- Nagyon kicsi effektus, nagyon nehéz kimutatni.

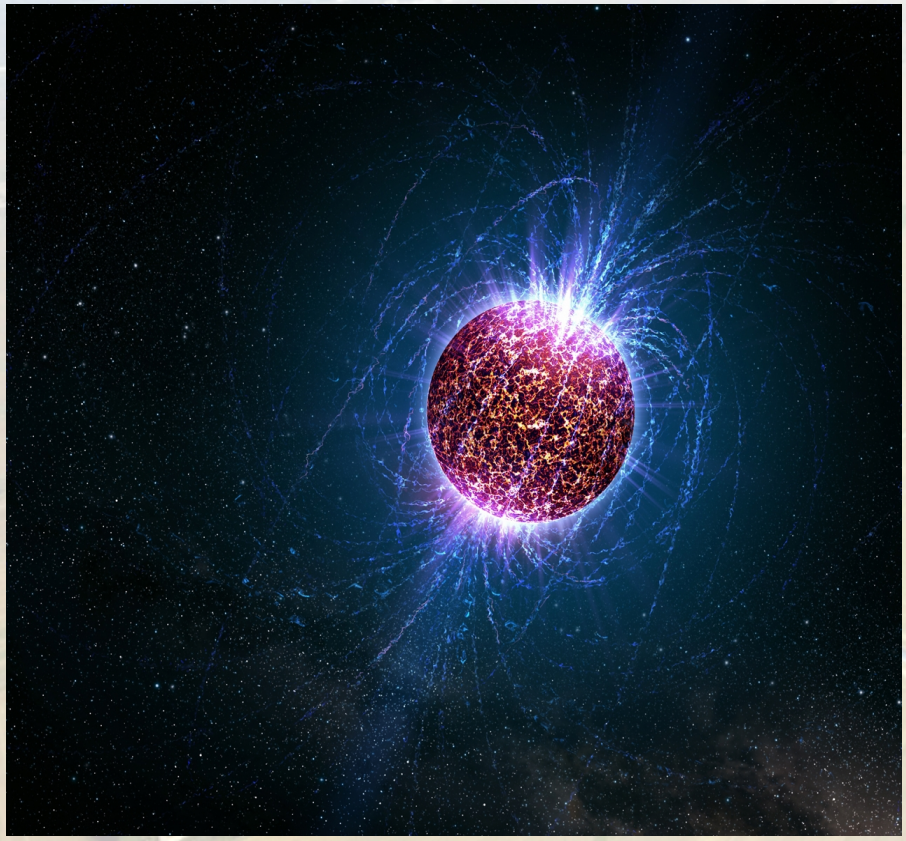


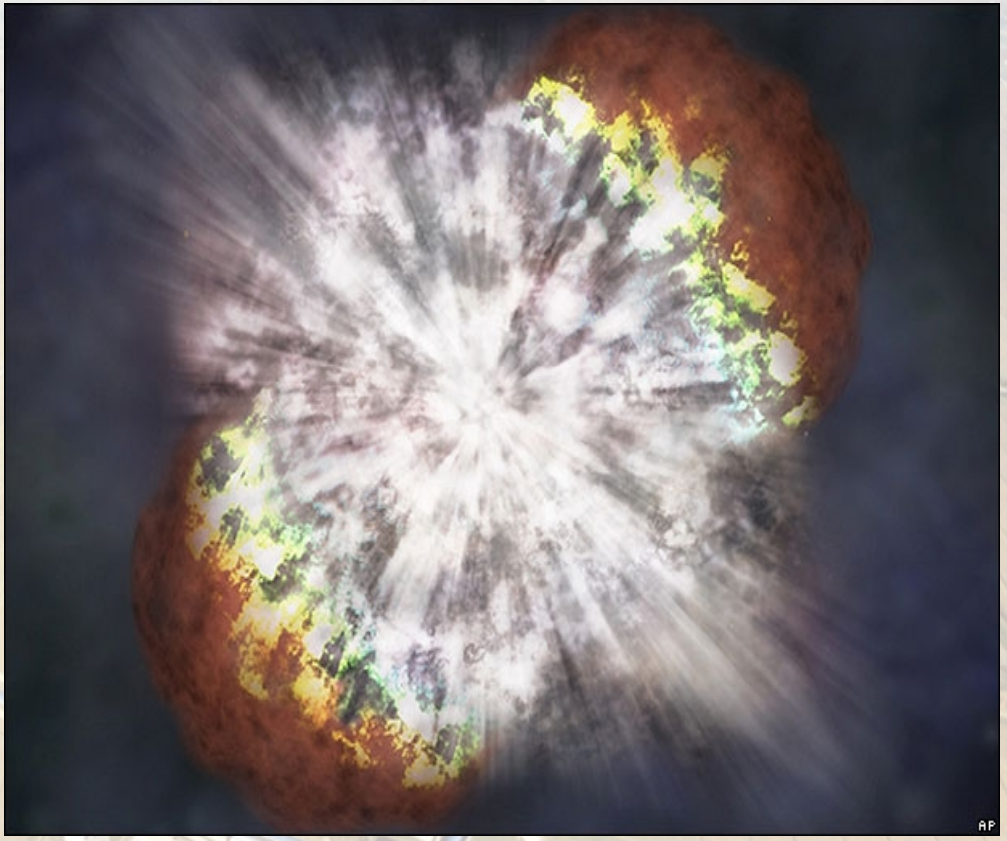




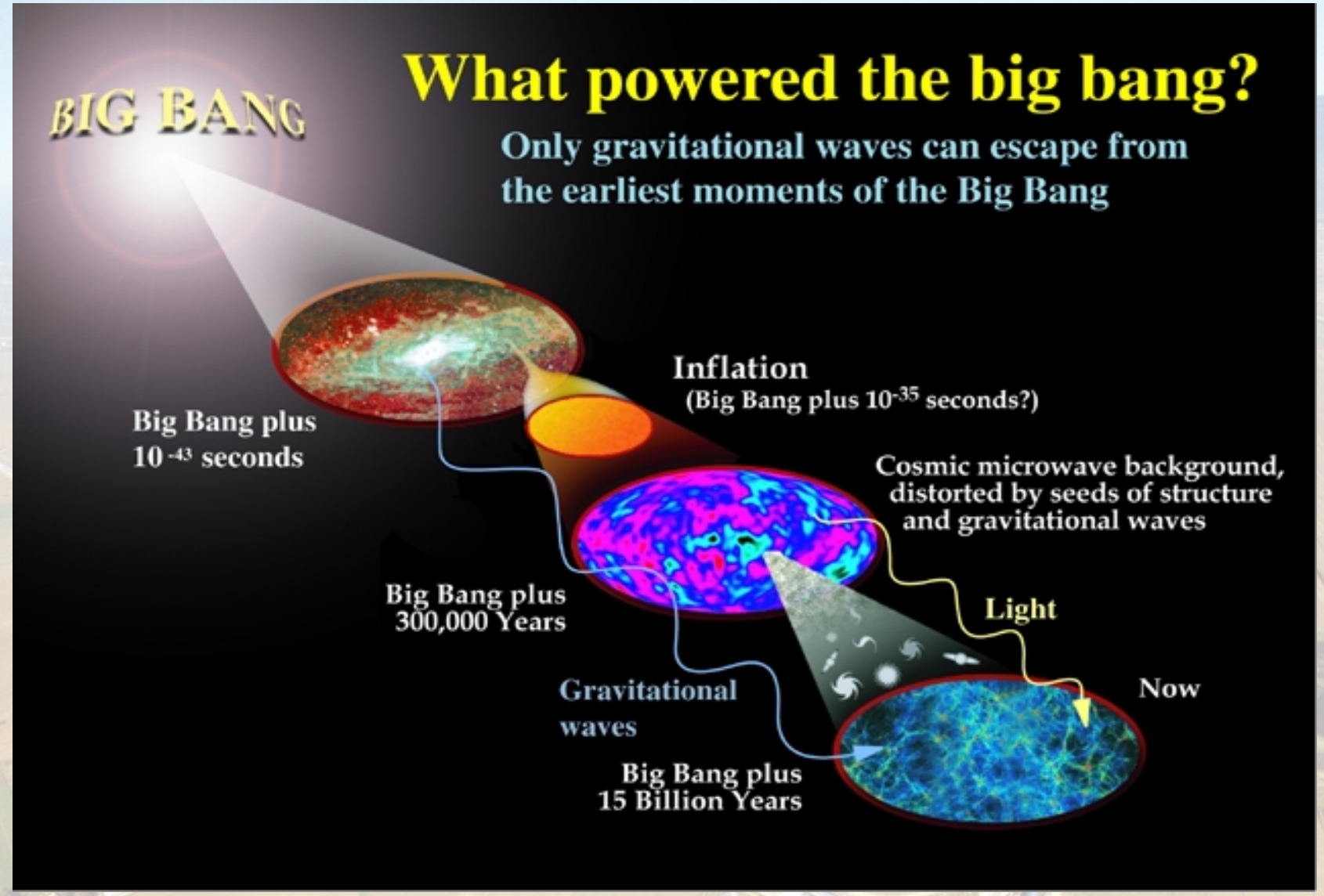
Gravitációs hullámforrások

- Forgó neutron csillagok
- Robbanásszerű jelenségek, szupernovák
- Az ősrobbanás környékéről visszamaradt gravitációs háttérsugárzás
- Bespirálozó csillagkettősök



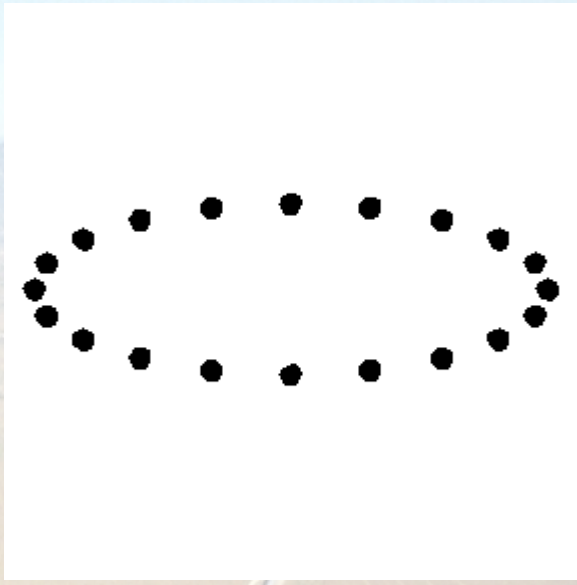


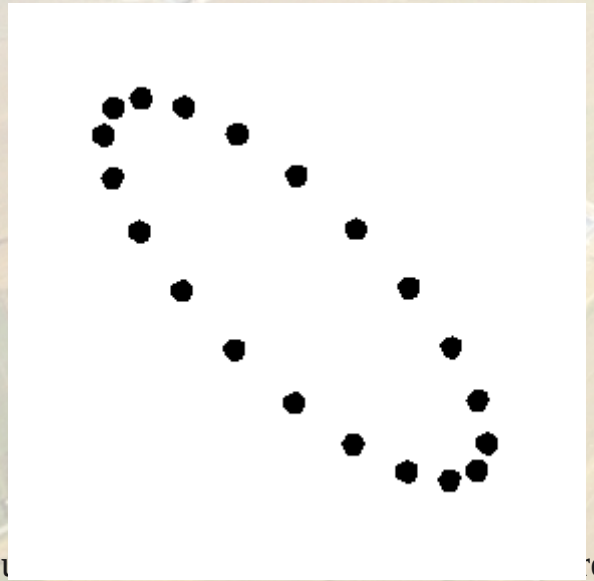
Gravitációs hullámforrások

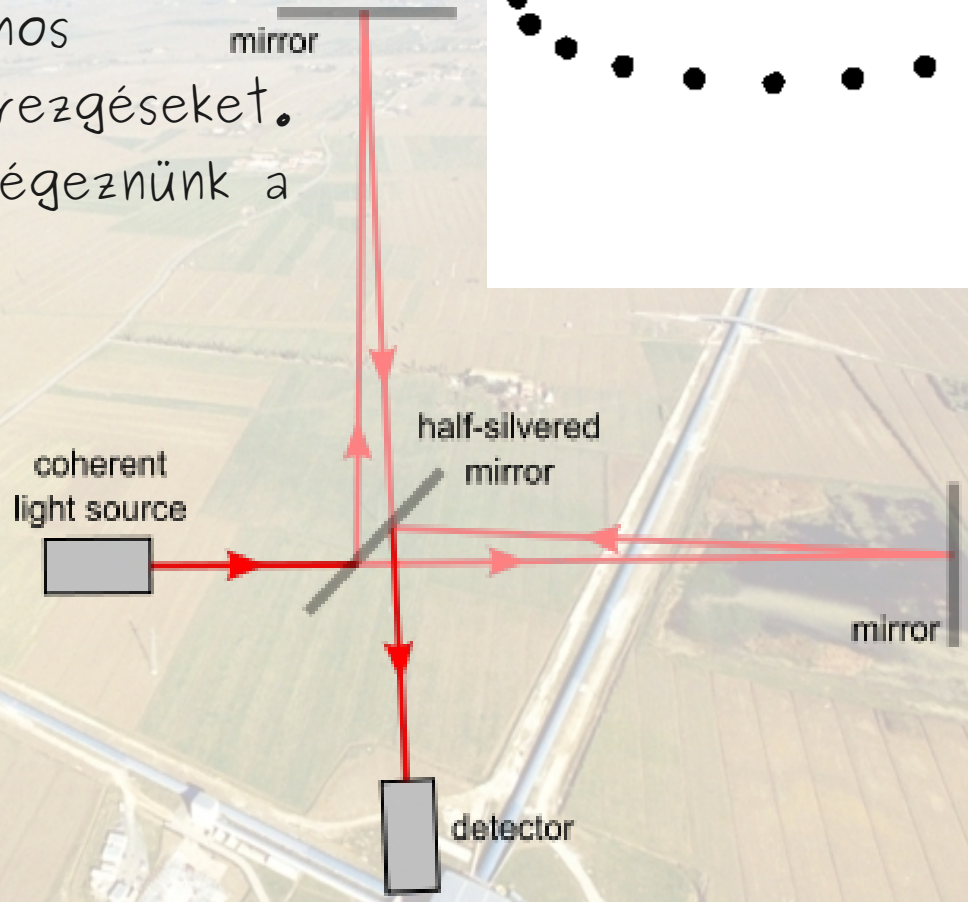


A gravitációs hullámokról

- A tér rezgései máshogy viselkednek más-más irányokban.
- Egyik irányban megrövidülnek a távolságok, a másik irányban meghosszabodnak.
- Két pont közötti távolság folytonos mérésével detektálhatjuk ezeket a rezgéseket.
- Nagyon nagy pontossággal kell elvégeznünk a kísérletet.

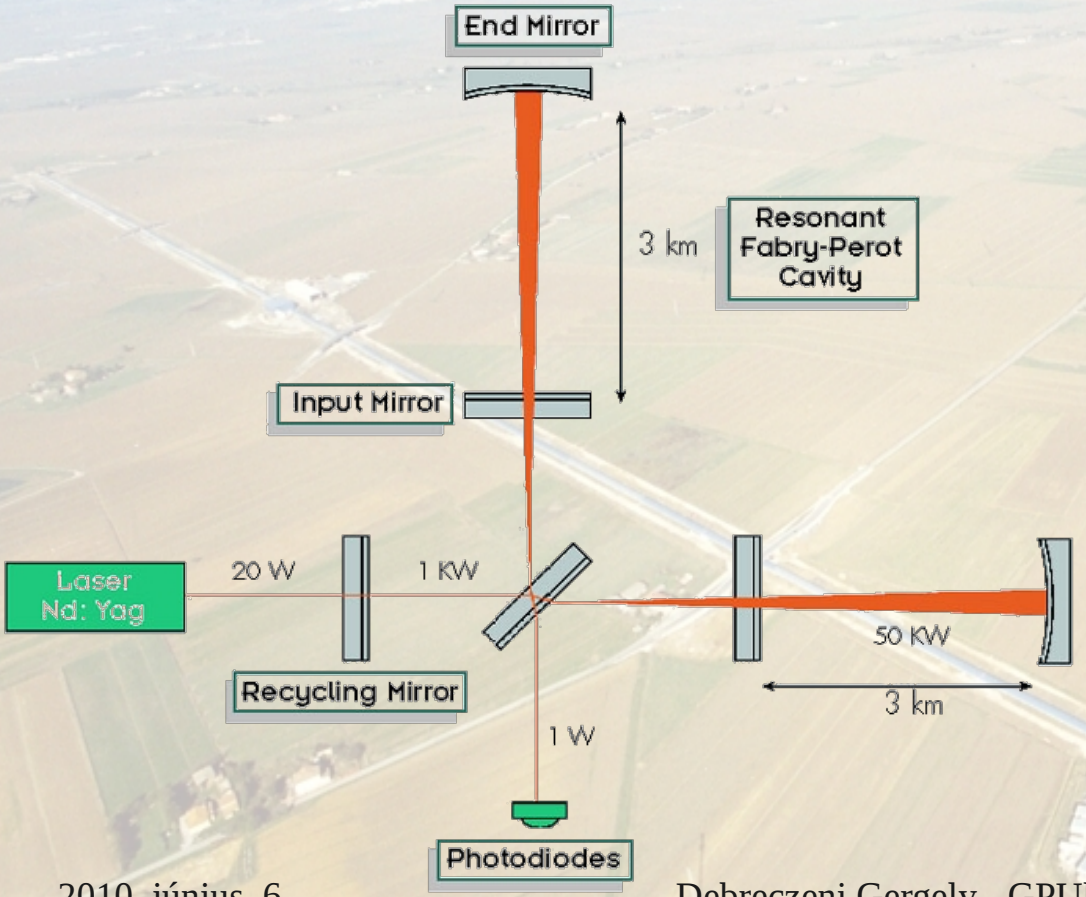






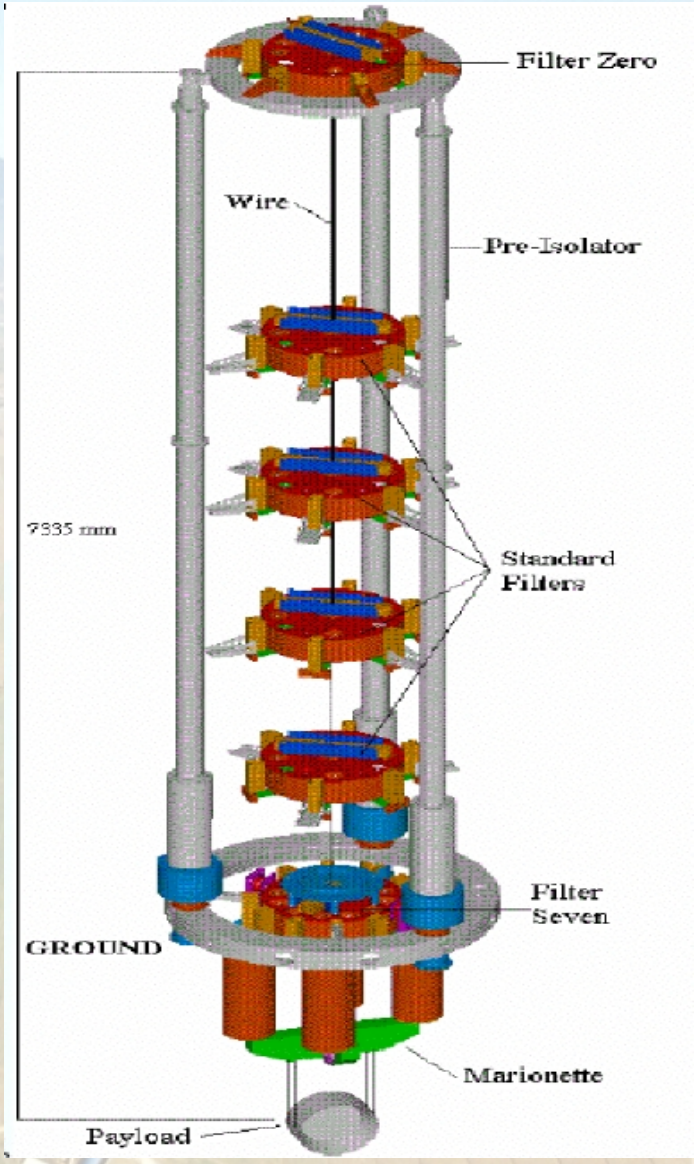
A Virgo kísérlet

- Cascina, Olaszország
- 3 km-es karhosszúság
- Fabry-Perot kavitások 50/150-es jósági tényező
- 6800 m^3, 10^{-10} mbar vákuum!!
- 20 W lézer
- stabil beton alapzat, 20-50 m mélyen
- 1 MW fogyaztás
- kiváló szeizmikus izoláció (10^{-5})



A Virgo kísérlet

- A mérés 10^{-18} m pontossággal képes távolságot mérni. Ez kisebb mint az atommag átmérője!
- Nagyon fontos a zavaró rezgések (akusztikus, szeizmikus, etc..) kiküszöbölése.
- A mérés kimenete egy h(t) függvény, egy idősor amely megadja a tükrök távolságát az idő függvényében.



Miért érdekesek a gravitációs hullámok ?

- A gravitációs hullámok felfedezése pontosan akkor segítséget jelentene a csillagászatban mint a röntgensugárzás használata az orvostudományban.
- A világegyetem és a csillagok egy teljesen új eddig még soha nem látott világába engedne betekintést !

OpenCl röviden

```
/********** Platform réteg létrehozása */

    err = clGetPlatformIDs(0, NULL, &numPlatforms);
    gclCheckError(err, "clGetPlatformIDs(numPlatforms)");

    if (0 < numPlatforms)
    {
        cl_platform_id* platforms = new cl_platform_id[numPlatforms];
        err = clGetPlatformIDs(numPlatforms, platforms, NULL);
        gclCheckError(err, "clGetPlatformIDs(platforms)");
        platform = platforms[0];
    }

   cl_context_properties cps[3] = {CL_CONTEXT_PLATFORM,
                                   (cl_context_properties)platform, 0};
    /* Use NULL for backward compatibility */
    cl_context_properties* cprops = (NULL == platform) ? NULL : cps;

/********** Context létrehozás, device hozzárendelés */

    cl_context context = clCreateContextFromType(cprops,
                                CL_DEVICE_TYPE_GPU, NULL, NULL, &err);
    gclCheckError(err, "clCreateContextFromType()");

    cl_device_id* devices;
    size_t devicenumber;
    clGetContextInfo(context, CL_CONTEXT_DEVICES, 0,
                                  NULL, &devicenumber);
    devices = (cl_device_id*)malloc(devicenumber);
    clGetContextInfo(context, CL_CONTEXT_DEVICES,
                                  devicenumber, devices, NULL);
    gclCheckError( devicenumber != 0 ? CL_SUCCESS : -1,
                                  "devicenumber <= 0");
```

```
//std::cout << devicenumber << '\n' << '\n';

/*************  Kernel forráskód betöltése */

std::ifstream file("tgen.cl");
gclCheckError(file.is_open() ? CL_SUCCESS : -1,
                                  "ifstream open ");

std::string prog( std::istreambuf_iterator<char>(file),
                                  (std::istreambuf_iterator<char>()));

/***************  Program létrehozás  */

cl_program program;
const char* kernelcode = prog.c_str();
size_t progsize = prog.size();
program = clCreateProgramWithSource(context, 1,
                                  &kernelcode, &progsize, &err);
gclCheckError(err, "clCreateProgramWithSource()");

/**************  Program fordítás */
   err = clBuildProgram(program, 0 , NULL/*devices*/,
                                  NULL, NULL, NULL);
   gclCheckError(err, "clBuildProgram");

/**************  Programból kernel */

    cl_kernel kernel;
    kernel = clCreateKernel(program, "prog1", &err);
    gclCheckError(err, "clCreateKernel()");
```

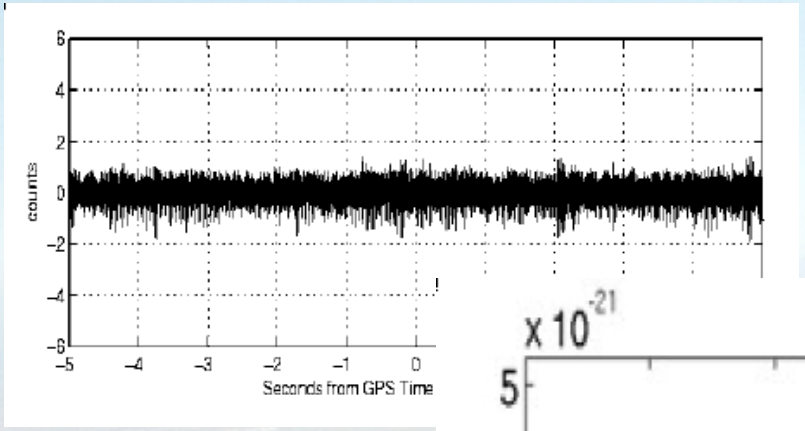
OpenCl röviden

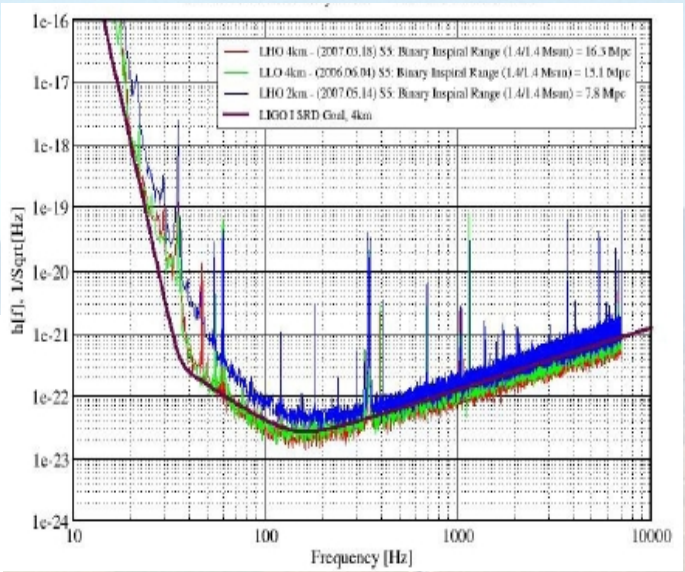
```
/********** Buffer létrehozása */

  buffers[0] = clCreateBuffer( context,
CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR,
sizeof(REAL)*templatesize, result, &err);
        gclCheckError(err, "buffers[0] = clCreateBuffer()");

        buffers[1] = clCreateBuffer( context,
CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR,
sizeof(REAL[1000]), result2, &err);
        gclCheckError(err, "buffers[1] = clCreateBuffer()");

/********** Kernel argumentumok */

        err = clSetKernelArg(kernel, 0, sizeof(buffers[0]),
&buffers[0]);
        gclCheckError(err, "kernel.setArg(0)");

        err = clSetKernelArg(kernel, 1, sizeof(REAL)*8,
NULL);
        gclCheckError(err, "kernel.setArg(1)");

/******** Parancs várakozási sor létrehozása */

        cl_command_queue commandQueue;
        commandQueue =
clCreateCommandQueue(context, devices[0], 0, &err);
        gclCheckError(err, "clCreateCommandQueue()");
```

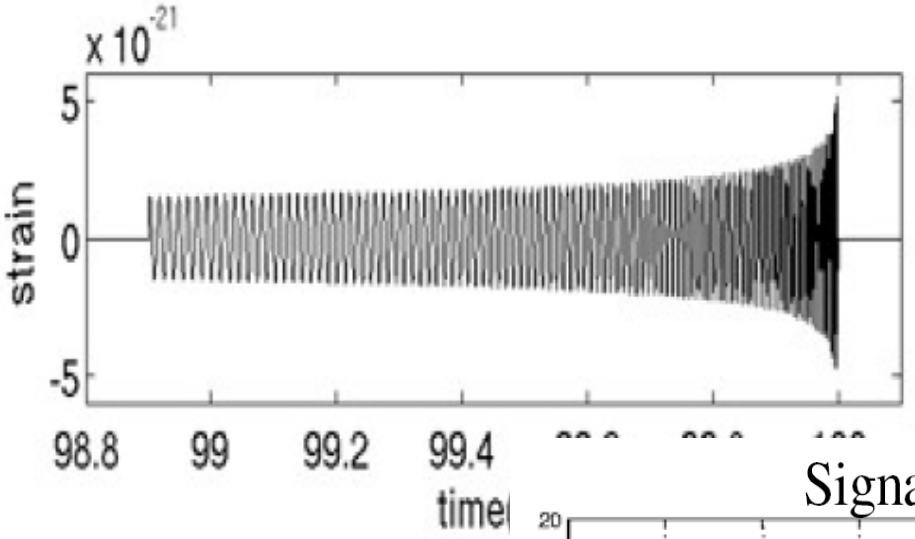
```
/********** Változók kiirása */

        err = clEnqueueWriteBuffer(commandQueue, buffers[0], CL_TRUE, 0,
sizeof(REAL)*templatesize, result, 0, NULL, &writeStat);
        gclCheckError(err, "clEnqueueWriteBuffer(writeQueue)");

/********* Kernel végrehajtása */

        err = clEnqueueNDRangeKernel( commandQueue, kernel,
globalWorkDim, 0, &globalWorkSize, &localWorkSize, 0, NULL,
&kernelStat);
        clWaitForEvents(1, &kernelStat);
        ftime(&gpute);
```

- Sokkal több munka
- Események (event-ek) a szinkronizáláshoz
- Aszinkron parancsvégrehajtás
- Kernel fordítás futási időben
- Több GPU - t és CPU -t is tud egyszerre kezelni
- Python, Ruby, C++, etc binding
- etc, etc...

Adatanalízis – SNR



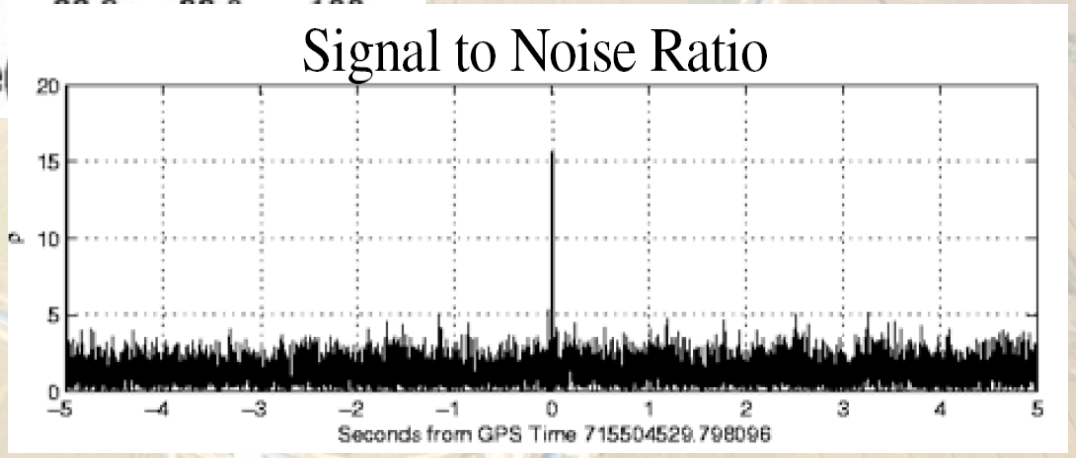




- SNR – Signal-to-Noise Ratio

$$(A,B) = \int \frac{A^*(f)B(f)}{S_n(f)} df,$$

$$\mathrm{SNR} = \frac{z}{\sqrt{\langle(\delta z)^2\rangle}} = \frac{(\tilde{Q},\tilde{s})}{\sqrt{(\tilde{Q},\tilde{Q})}},$$



Adatanalízis – FFT

- FFT – Fast Fourier transofrmation – mindenütt használják
- Az SNR számításhoz is szükség van FFT-re
- Ha az FFT számítást GPU-n végezzük el a teljes futási idő negyedére csökken
- FFT implementáció létezik CUDA-ban és OpenCL-ben is , nagyon könnyű az implementáció !

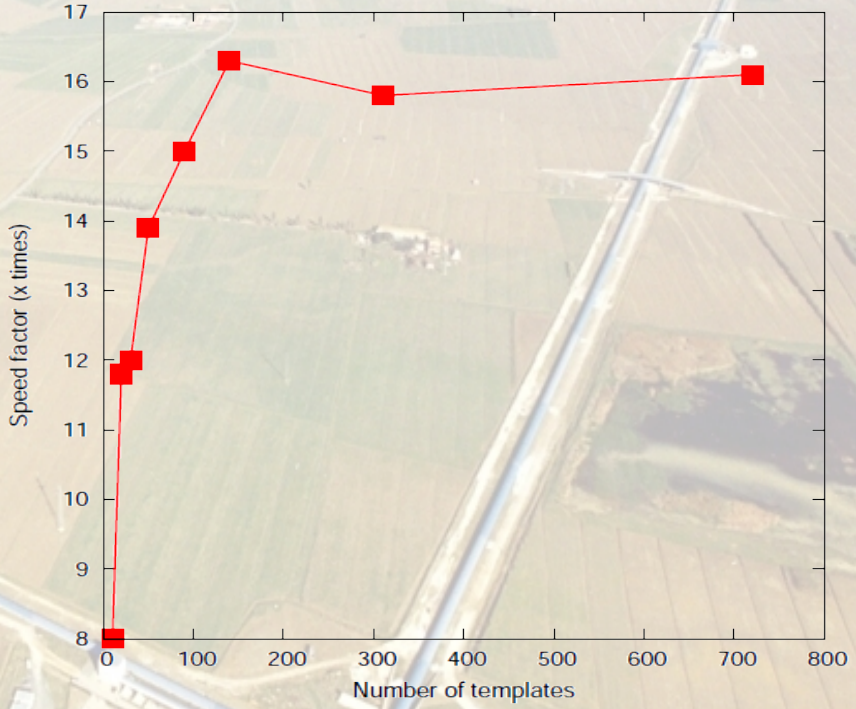
Adatanalízis – SNR

- SNR – Signal-to-Noise Ratio lényegében komplex vektorok skaláris szorzata
- Triviális módon párhuzamosítható, CUDA-ban is és OpenCL-ben is létezik BLAS csomag

Adatanalízis – Chi²

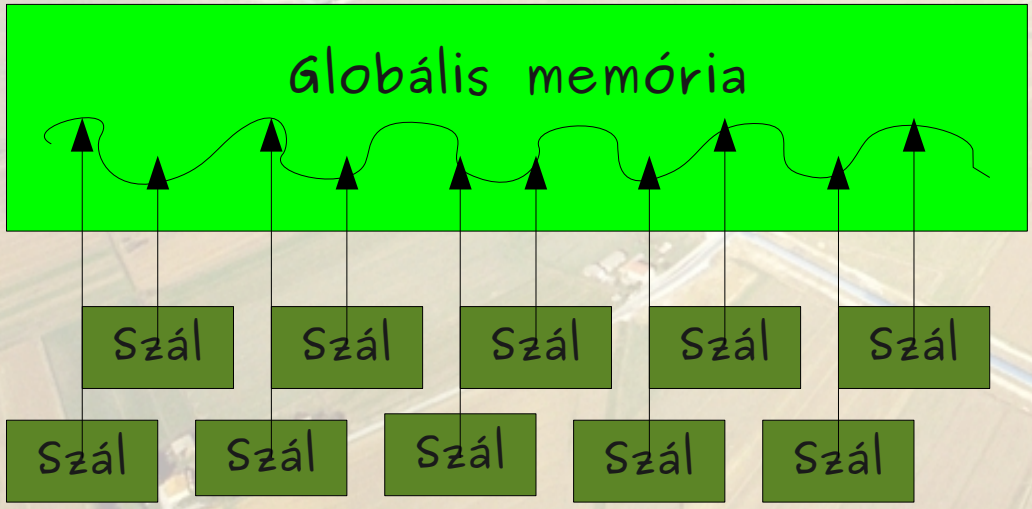
- Az SNR számítást nem csak egyszer végezzül el, hanem a template-t elosztjuk 16 részre és minden egyes szeletben kiszámoljuk az SNR-t.
- Az így kapott SNR-ok chi2 eloszlást követnek.
- Nagyon jó teszt a zajok 'glitchek' ellen !
- A 16 SNR számolás, a 16 FFT párhuzamosítható a GPU-n, egyszerre végrehajtható !

16x



Adatanalízis – Templétek

- Nagyszámú elméleti hullámforma templétet kell generálnunk, amelyek különböző fizikai konfigurációknak felelnek meg.
- A templétek paramétertérbeli távolsága függ az aktuálisan mért idősor speltrális sűrűségfüggvényétől is.
- Több 100K templét generálása szükséges.
- 4 khz mintavételezési frekvencio, 16 sec hosszú templétek, 65536 érték kiszámítása szépen kiosztható a GPU processzorok között !
- A templétek számításigényes függvényeket (sin, cos, pow) tartalmaznak, ezért a gyorsulás kisebb a vártnál...



30x

Adatanalízis – Templétek

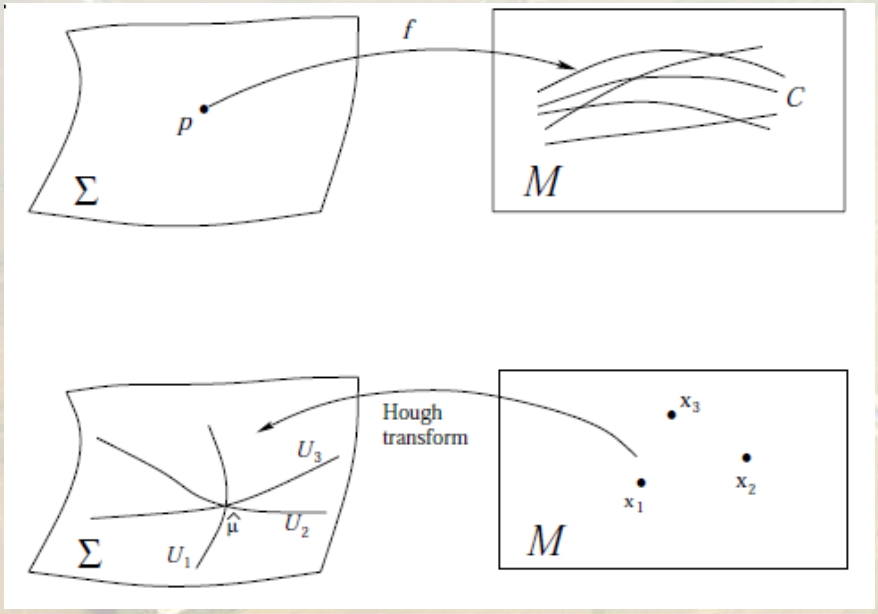
```
#define REAL float;
REAL t;
for (int i = 0; i < templatesize; i++) {
    t = i/samplerate;
    template[i] = cpufunc(t);
}
```

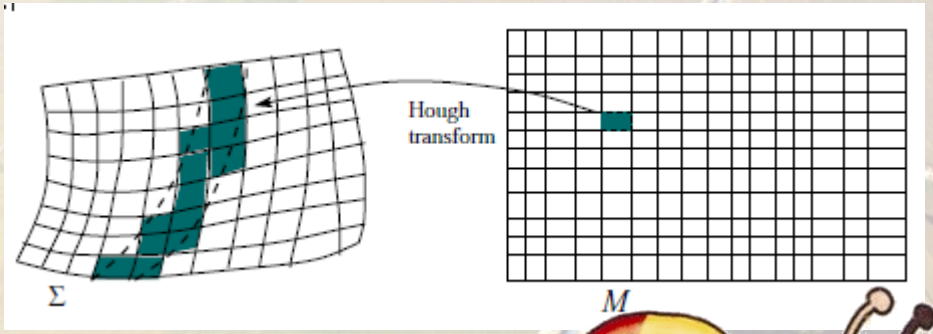
- Triviális kód CPU-n
- GPU – n több kódolás, de gyorsabb
- Vektorprovesszorok használata (float4)
- Globális memoriába irás drága lassu művelet

```
#pragma OPENCL EXTENSION cl_khr_fp64: enable
#define REAL float4

REAL gpufunc(REAL t) {
  return template(t);
}
__kernel void prog1(__global REAL * template,
                    __local REAL * temp)
{

    int   idg = get_global_id(0);
    int   min, max;
    REAL  t;
    const REAL  samplerate   = {4096, 4096, 4096,4096};
    const REAL  shift        = {1,2,3,4};
    const REAL  ten          = {10.,10.,10.,10.};
    min = idg * 256;
    max = min + 256;
    for (int i = min; i < max; i += 4) {
      t = (shift + i)/samplerate;
      template[i] = gpufunc(t);
     }
}
```

Adatanalízis – Hough transz.

- A Hough transzformáció egyre elterjedtebb képfeldolgozási módszer.
- Mintafelismerés, csökkentett bemenő paraméter esetén.
- A gravitációs hullám analízisben az ido – frekvencia síkot osztjuk fel kis négyzetekre, és csak azokat tartjuk meg ahol az intenzitás elér egy küszöbértéket.
- Ez a 2 dimenziós probléma szintén jól párhuzamosítható GPU-n





20x

Elmélet - Kontrakció

- Nemlineáris dinamikus rendszerek időfejlődésének vizsgálata. A terek R^2 x K -n vannak kifejtve (K egy 2D kompakt sokaság). Radiális és időfejlődés R^2-ben , a maradék az ortogonális Laplace gömbfüggvények szerint K-n kifejtve. A terek K-n vett pontszerű szorzatainál a lép fel a gömbfüggvények együtthatói kiszámításának problémája:

$$C_k = M_{ijk} * A_i * B_j$$

- A,B és C kb. 1000 elemű komplex vektorok.
- M valós ritka mátrix, azaz kb csak 200K nem nulla eleme van.
- A feladat jól párhuzamosítható, M-et ritka mátrix formátumban tárolva, és a k index szerinti munkacsoportokat kialakítva, jelentős gyorsulást érhetünk el.
- Esetünkben minde 'double' , ezért nem optimális a kártya számára.

7x-30x

Tényleg szükséges ez ?

- Alapvető kérdés, hogy tényleg szükség van-e GPU-s gyorsításra !
- A kísérleti gravitációfizikában:
  - On-line , szimultán elektromágneses és gravitációs megfigyelések triggeléréséhez nagyon fontos.
  - Költséghatékonyság

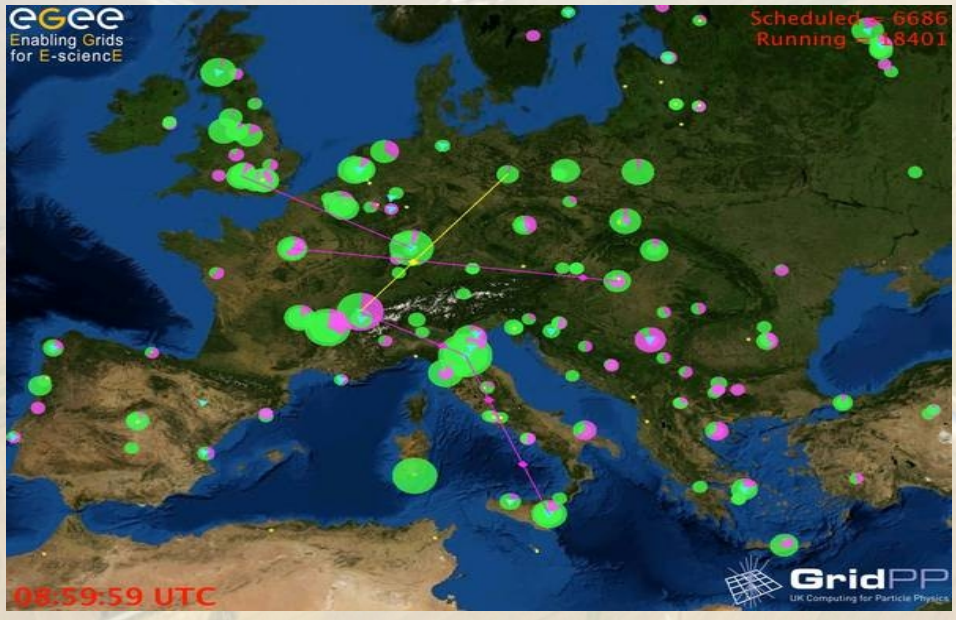
Van még olyan adat-analízis módszer aminek számításikapacitás igénye nagyságrendekkel meghaladja a napjainkban elérhetőt. A GPU-k és a Many Core Computing segítségével ez 5-6 éven belül elérhetővé válhat és megteremtheti a reális esélyt a gravitációs hullámok felfedezésre !

A számítógépek lesznek a 21 század detektorai, felfedezői.

Igen

Grid és GPUk, avagy HPC contra Grid computing

- EGEE Grid, LHC, HunGrid
- Több 100K PC, 5 kontinens, 120 PByte tárhely, 300+ intézet, 10K+ kutató, 150+ kutatócsoport
- Az RMKI-ban is fejlesztjük
- GPU gyorsított Grid node-ok a Grid információs rendszerében meghirdetve, HPC programok a Griden !



http://virgo.rmki.kfki.hu

Az RMKI Virgo csoport

http://virgo.rmki.kfki.hu