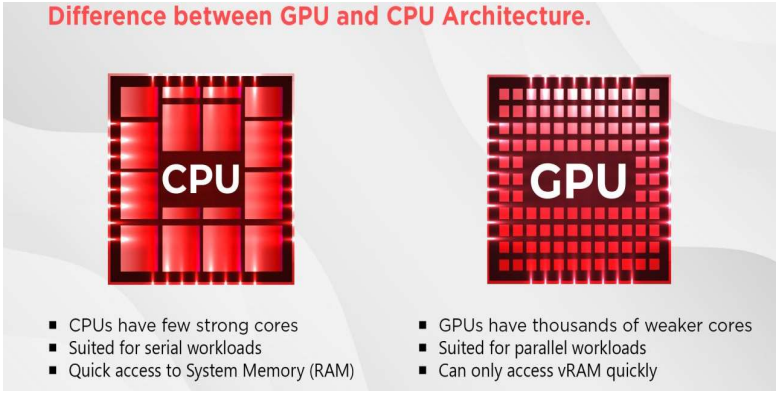
پیاده سازی با استفاده از منابع کارت گرافیک در دات نت

خیلی دوست داشتم یک روش ساده پیدا کنم که جدا از پیچیدگی ها مرسوم فریم ورک های استفاد ها از gpu ها بتوانم کدهای ساده ام را به سریعترین حالت روی گرافیک اجرا بگیرم. که خوب دیدم راه حل های جالبی هست. یکی از این راه حل ها که من خیلی حال کردم و خوب ازش استفاده کردم. ILGPU بود که برای فریم ورک و ماشین دات نت طراحی و پیاده سازی شده.



چندتا نکته کوچیک :

ما نمی توانیم GPU را مانند یک CPU در پیاده سازی در نظر بگیریم، در واقع در این سه حوزه باید مقایسه کرد:

- دسترسی به حافظه
- محلی بودن داده
- Threading

در واقع GPU یک CPU نیست. (GPU یک مجتمع از هزار یا هزاران هسته با اشتراک چند ده یا صدها واحد کنترل و کش است ولی CPU یک یا حداکثر چند ده هسته هست با همان تعداد واحد کنترل و کش و داستان TPU که جدا و اختصاصی برای یادگیری ماشین هست با معماری ماتریسی و سیستولی)

CPU چگونه کار می کند؟

یک پردازنده سنتی یک چرخه بسیار ساده دارد: واکشی، رمزگشایی، اجرا.

یک دستورالعمل را از حافظه می گیرد (واکشی)، نحوه اجرای دستورالعمل مذکور (رمزگشایی) را مشخص می کند و دستور را انجام می دهد (اجرا). سپس این چرخه برای تمام دستورالعمل های الگوریتم شما تکرار می شود. اجرای این جریان خطی از دستورالعمل ها برای اکثر برنامه ها خوب است زیرا CPU ها بسیار سریع هستند و اکثر الگوریتم ها به صورت سریالی هستند.

وقتی الگوریتمی دارید که می تواند به صورت موازی پردازش شود چه اتفاقی می افتد؟ یک CPU چندین هسته دارد که هر کدام واکشی، رمزگشایی و اجرای خود را انجام می دهند. می‌توانید

الگوریتم را در تمام هسته‌های CPU پخش کنید، اما در نهایت هر هسته همچنان یک جریان دستورالعمل‌ها را اجرا می‌کند، احتمالاً همان جریان دستورالعمل‌ها، اما با داده‌های متفاوت.CPU ها ترفندی برای برنامه های موازی به نام SIMD دارند. اینها مجموعه‌ای از دستورالعمل‌ها هستند که به شما اجازه می‌دهند تا یک دستور عملیات را روی چندین قطعه داده به طور همزمان انجام دهد. (بحث بیشتر مختص درس معماری کامپیوترهاست که بسیار مفصل است.) کتاب خانه و فریم ورک های زیادی به ما کمک می کنند، که اکثرشان تبدیل و دستری به کدهای C++ را پیشنهاد می کنند. من قبلا برای کارهای گرافیکی از کتابخانه SHARPDL استفاده کرده بودم. وای الان منظور کارهای گرافیکی نیست و اجرای کد روی پردازنده گرافیکی برام موضوعیت داره.

لیست این ها عبارتند از :

CUDAfy .NET allows easy development of high performance GPGPU applications completely from the .NET. It's developed in C#. (by lepoco)

arrayfire-rust - Rust wrapper for ArrayFire

NvAPIWrapper - NvAPIWrapper is a .Net wrapper for NVIDIA public API, capable of managing all aspects of a display setup using NVIDIA GPUs

novideo_srgb - Calibrate monitors to sRGB or other color spaces on NVIDIA GPUs, based on EDID data or ICC profiles

cuda-api-wrappers - Thin C++-flavored header-only wrappers for core CUDA APIs: Runtime, Driver, NVRTC, NVTX.

TinyNvidiaUpdateChecker - Check for NVIDIA GPU driver updates!

srmd-ncnn-vulkan - SRMD super resolution implemented with ncnn library

ZenTimings A free, simple and lightweight app for monitoring memory timings on Ryzen platform.

Amplifier.NET - Amplifier allows .NET developers to easily run complex applications with intensive mathematical computation on Intel CPU/GPU, NVIDIA, AMD without writing any additional C kernel code. Write your function in .NET and Amplifier will take care of running it on your favorite hardware.

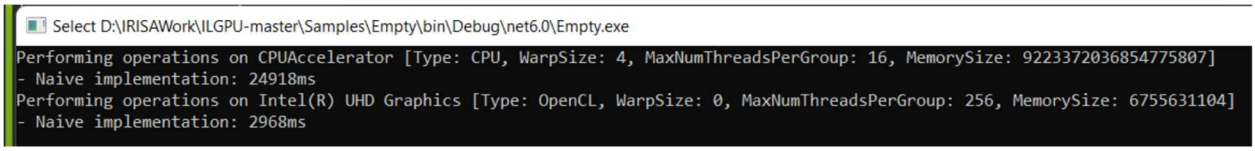
Hybridizer - Examples of C# code compiled to GPU by hybridizer

Hades-VegaM - Moded AMD RX Vega M series gpu driver to support Intel NUC Hades Canyon and all other products on Windows 10/11

waifu2x-converter-cpp - Improved fork of Waifu2X C++ using OpenCL and OpenCV

کتابخانه ساده ای که من استفاده کردم و خیلی ساده بود کتابخانه ILGPU بود . نتیجه واقعا روی سیستم فوق العاده بود.یک برنامه اعداد اول نوشتم که به صورت عمل می کند.(۲ تا ۱۰،۰۰۰)

از عدد ۲ تا یک کمتر از عدد اگر قابل بخش پذیری نبود، عدد اول است. (قصدم تست بود و نه بهینه کار کردن)



برنامه تست را روی گیت هاب به ادرس:

https://github.com/rajaei/ILGPUTest

برنامه نویسی بسیار ساده ای دارد. کافی است بعد از نوشتن روال، انرا به تابع کرنل پاس بدهید و تمام.

```
using ILGPU;
using ILGPU.Runtime;
using ILGPU.Runtime.Cuda;
using System.Diagnostics;
// گرفتن یک زمینه پیش فرض برای شروع کار

using var context = Context.CreateDefault();
// به ازای هر دستگاه موجود یک بار برنامه را اجرا می کنیم.

foreach (var device in context.Devices)
{
    using var accelerator = device.CreateAccelerator(context);
    Console.WriteLine($"Performing operations on {accelerator}");
    var sw = new Stopwatch();
    sw.Restart();
    MyKernel(accelerator);
    sw.Stop();
    Console.WriteLine($"- Naive implementation: {sw.ElapsedMilliseconds}ms");
}
Console.ReadLine();
// روال اصلی برنامه که محاسبه عدد اول را دارد

static void MyCalcute(Index1D total, ArrayView<long> results, Input<int> TotalTag)
{
    int Total = 0;
    for (int i = 1; i < 10000; i++) {
        bool isPrime = true;
        for (int j = 2; j < i - 1; j++)
            if ((i % j) == 0) {
                isPrime = false;
                break;
            }
        if (isPrime)  {
            results[Total] = i;
            Total++;
        }
    }
}
// پاس دادن روال به کرنل دستگاه

static void MyKernel(Accelerator accelerator)
{
    using var buffer = accelerator.Allocate1D<long>(10000);
    var kernel = accelerator.LoadAutoGroupedStreamKernel<Index1D, ArrayView<long>,
Input<int>>(MyCalcute);
    kernel(
        (int)buffer.Length,
        buffer.View,
        10000);
    var results = buffer.GetAsArray1D();
}
```

متفاوت است ولی، نگران نباشید. کافی ایست این افزونه را درصورتی که CUDAالبته این با پردازنده گرافیم شما پشتیبانی می کند دانلود کنید و سپس با تغییر کوچیک از اون روش استفاده کنید.

```
using var context =
context.GetCudaDevices()  به جای  Context.CreateDefault();
```

هدفم صرفا شراکت شما در حال برده شده از این کار بود. طبعا دانش و اطلاعات بیشتری باید در این زمینه داشت برای کارهای بزرگتر.

رفرنس ها:

- https://alisterta.github.io/۲۰۱۸-۰۹-۰۳/TPU-%DA%۸۶%DA%AF%D۹%۸۸%D۹%۸۶%D۹%۸۷-%DA%A۹%D۸%A۷%D۸%B۱-%D۹%۸۵%DB%۸C%DA%A۹%D۹%۸۶%D۸%AF
- https://dataio.ir/%D۸%AA%D۹%۸۱%D۸%A۷%D۹%۸۸%D۸%AA-cpu-gpu-tpu-lolrppjxzxpl
- https://www.ilgpu.net/docs/ReadMe
- https://www.libhunt.com/compare-CUDAfy.NET-vs-ILGPU?ref=compare

مسعود رجایی – Mr.rajjaei@gmail.com

/https://www.linkedin.com/in/masoudrajaei