

NVIDIA nvprof / nvvp工具安装和使用介绍

原创

TracelessLe

于 2020-12-08 20:39:17 发布

阅读量1.4w

收藏 30

点赞数 10

版权

分类专栏：

GPU加速

CUDA

文章标签：

linux

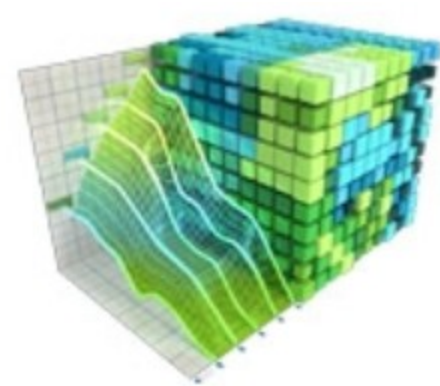
nvidia

cuda

gpu

nvvp

前言



NVIDIA nvprof / nvvp工具是英伟达N卡GPU编程中用于观察的利器。全称是NVIDIA Visual Profiler，是由2008年起开始支持的性能分析器。交互性好，利于使用。其中记录运行日志时使用命令nvprof，可视化显示日志时使用命令nvvp。

该工具的官方介绍如下：

NVIDIA Visual Profiler

This is a cross-platform performance profiling tool that delivers developers vital feedback for optimizing CUDA C/C++ applications. First introduced in 2008, Visual Profiler supports all CUDA capable NVIDIA GPUs shipped since 2006 on Linux, Mac OS X, and Windows.

<https://blog.csdn.net/TracelessLe>

不过在最近几年，英伟达官方推出了新的性能分析工具NSight，官方更加建议使用新的工具，给出的原因是NSight运行时消耗的资源更少，统计的数据更加贴近实际运行情况的数据。相比之下使用nvprof/nvvp方式运行时消耗资源较多，数据统计容易不准确。

Profiler User's Guide

The user manual for NVIDIA profiling tools for optimizing performance of CUDA applications.

Profiling Overview


This document describes NVIDIA profiling tools that enable you to understand and optimize the performance of your CUDA, OpenACC or OpenMP applications. The [Visual Profiler](#) is a graphical profiling tool that displays a timeline of your application's CPU and GPU activity, and that includes an automated analysis engine to identify optimization opportunities. The [nvprof](#) profiling tool enables you to collect and view profiling data from the command-line.

Note that Visual Profiler and nvprof will be deprecated in a future CUDA release. The NVIDIA Volta platform is the last architecture on which these tools are fully supported. It is recommended to use next-generation tools [NVIDIA Nsight Systems](#) for GPU and CPU sampling and tracing and [NVIDIA Nsight Compute](#) for GPU kernel profiling.

Refer the [Migrating to Nsight Tools from Visual Profiler and nvprof](#) section for more details.

<https://blog.csdn.net/TracelessLe>

除此以外，英伟达还给出了其他的工具，如果希望深入GPU编程，那么这些工具都是傍身利器。

 **NVIDIA DEVELOPER**

HOME BLOG NEWS FORUMS DOCS DOWNLOADS TRAINING

ACCOUNT

Performance Analysis Tools

[Home](#) > [High Performance Computing](#) > [Tools & Ecosystem](#) > [Performance Analysis Tools](#)

NVIDIA Nsight Systems

NVIDIA® Nsight™ Systems is a system-wide performance analysis tool designed to visualize application's algorithm, help you select the largest opportunities to optimize, and tune to scale efficiently across any quantity of CPUs and GPUs in your computer; from laptops to DGX servers.

NVIDIA® Nsight™

The ultimate development platform for heterogeneous computing. Work with powerful debugging and profiling tools, optimize the performance of your CPU and GPU code. Find out about the Ecilipse Edition and the graphics debugging enabled Visual Studio Edition.

NVIDIA Visual Profiler

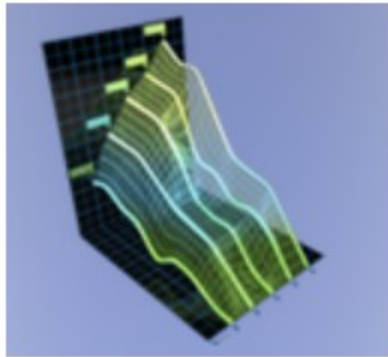
This is a cross-platform performance profiling tool that delivers developers vital feedback for optimizing CUDA C/C++ applications. First introduced in 2008, Visual Profiler supports all CUDA capable NVIDIA GPUs shipped since 2006 on Linux, Mac OS X, and Windows.

TAU Performance System® This is a profiling and tracing toolkit for performance analysis of hybrid parallel programs written in CUDA, and pyCUDA., and OpenACC.	VampirTrace A performance monitor which comes with CUDA, and PyCUDA support to give detailed insight into the runtime behavior of accelerators. Enables extensive performance analysis and optimization of hybrid programs.	The PAPI CUDA Component A hardware performance counter measurement technology for the NVIDIA CUDA platform which provides access to the hardware counters inside the GPU. Provides detailed performance counter information regarding the execution of GPU kernels.
The NVIDIA CUDA Profiling Tools Interface (CUPTI) provides performance analysis tools with detailed information about GPU usage in a system. CUPTI is used by performance analysis tools such as the NVIDIA Visual Profiler, TAU and Vampir Trace.	NVIDIA Topology-Aware GPU Selection (NVTAGS) is a toolset for HPC applications that enables faster solve times with high GPU communication-to-application run-time ratios. NVTAGS intelligently and automatically assigns GPUs to message passing interface (MPI) processes, thereby reducing overall GPU-to-GPU	

安装与使用

Linux & Windows

在带有N卡的Linux和Windows机器环境下，在安装好CUDA Toolkit后则自带了nvprof / nvvp等工具。且一般会有相应的应用图标（NVIDIA Visual Profiler 和 NSight），可以直接点击图标打开。



也可以通过命令行方式打开。

```
2019 bin2c*
2019 computeprof -> nvvp*
2020 crt/
2019 cudafe++*
2019 cuda-gdb*
2019 cuda-gdbserver*
2019 cuda-install-samples-10.2.sh*
2019 cuda-memcheck*
2019 cuobjdump*
2019 fatbinary*
2019 nsight*
2019 nsight_ee_plugins_manage.sh*
2019 nsight-sys*
2019 nsys*
2019 nsys-exporter*
2019 nvcc*
2019 nvcc.profile
2019 nvdisasm*
2019 nvlink*
```




以Linux系统命令行方式为例，说明使用方法。

直接打印跟踪日志

```
1 | $ nvprof --print-gpu-trace python test.py
```

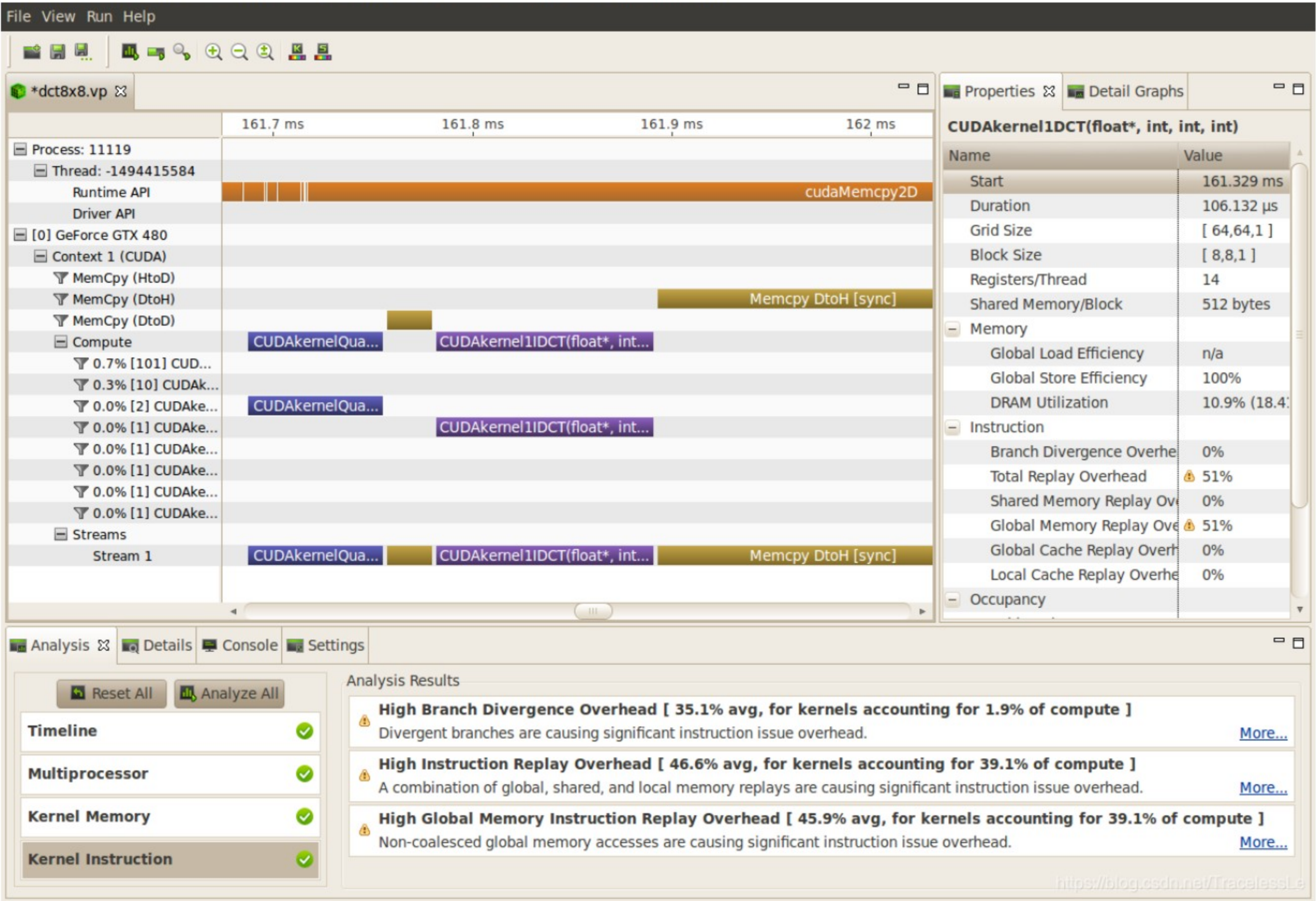
生成nvvp日志文件

```
1 | $ nvprof -o prof_name.nvvp python test.py
```

打开nvvp日志文件

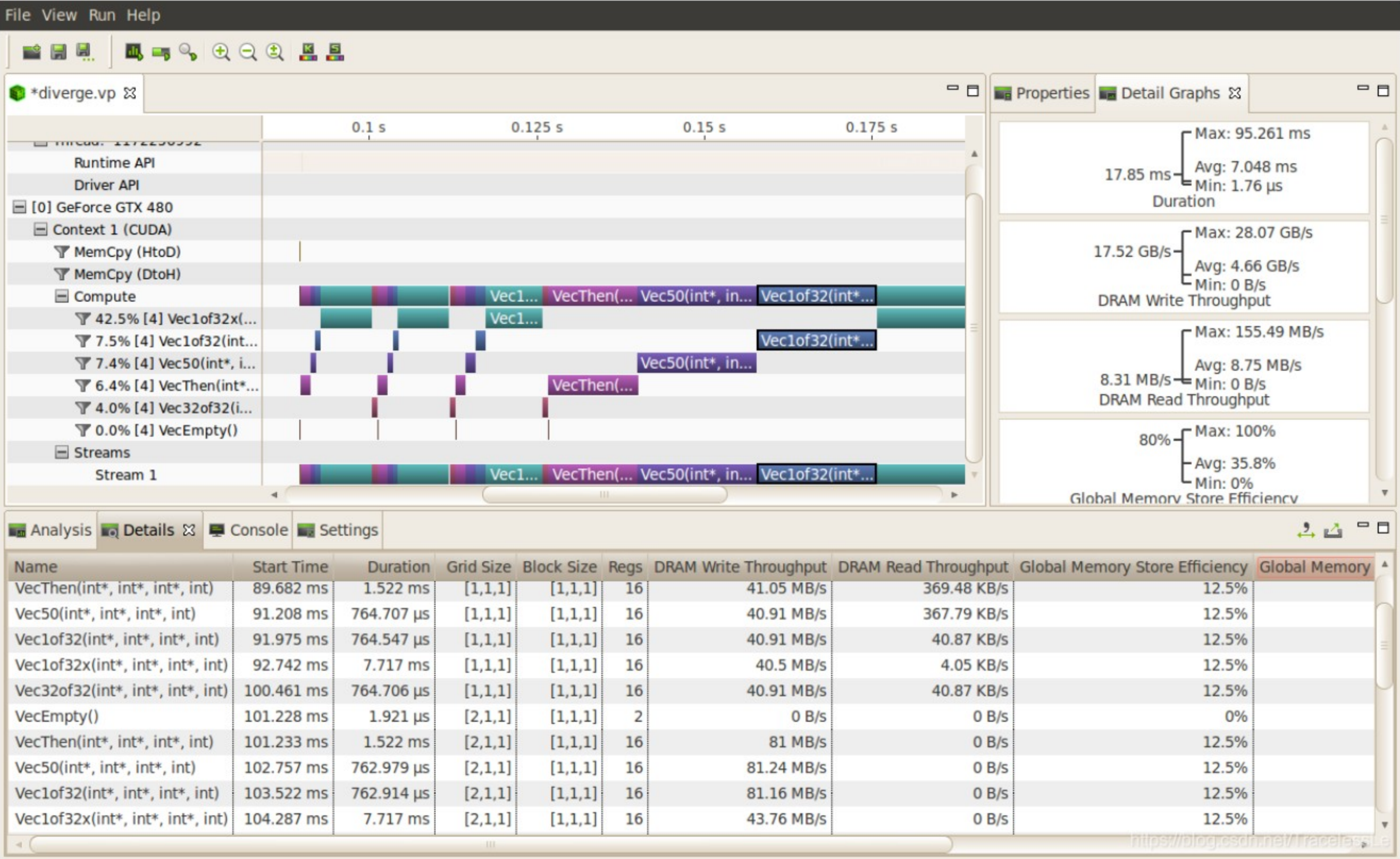
```
1 | $ nvvp prof_name.nvvp
```

最开始打开日志时显示如下，可以初步看到整个程序运行时的耗时情况：



X轴是时间轴，表示了程序运行的时间。Y轴分为两个大块，包括进程号模块和显卡卡号模块。其中进程号下又细分为线程模块，其下又分为Runtime API和Driver API耗时。而显卡模块下分为Context模块，其下又细分为MemCpy、Compute和Stream模块。MemCpy展示了显存拷贝的耗时情况，Compute展示了实际CUDA Kernel运行时消耗，Stream展示了流上的时间消耗情况。需要注意的是子模块的信息是可以投射回父模块的时间轴上的。可以通过放大缩小来查看某个关注周期内的

时间消耗情况。



一般在使用时，如果是多个循环，可以先找出某个代表性循环周期，在时间轴上标记处这一周期，然后再仔细查看Compute部分的耗时情况，保证时间消耗不要浪费在内存开辟和释放上，主要花在核函数运行上。优良的CUDA程序从Stream轴看起来是比较连续的，不会出现大量的空白区间。另外还可以关注某些大量耗时的核函数，考虑优化核函数的设计。

nvprof和nvvp还有更多的特性，例如多进程跟踪支持，核函数调用和运行情况统计等。详细使用可以参考官方文档。

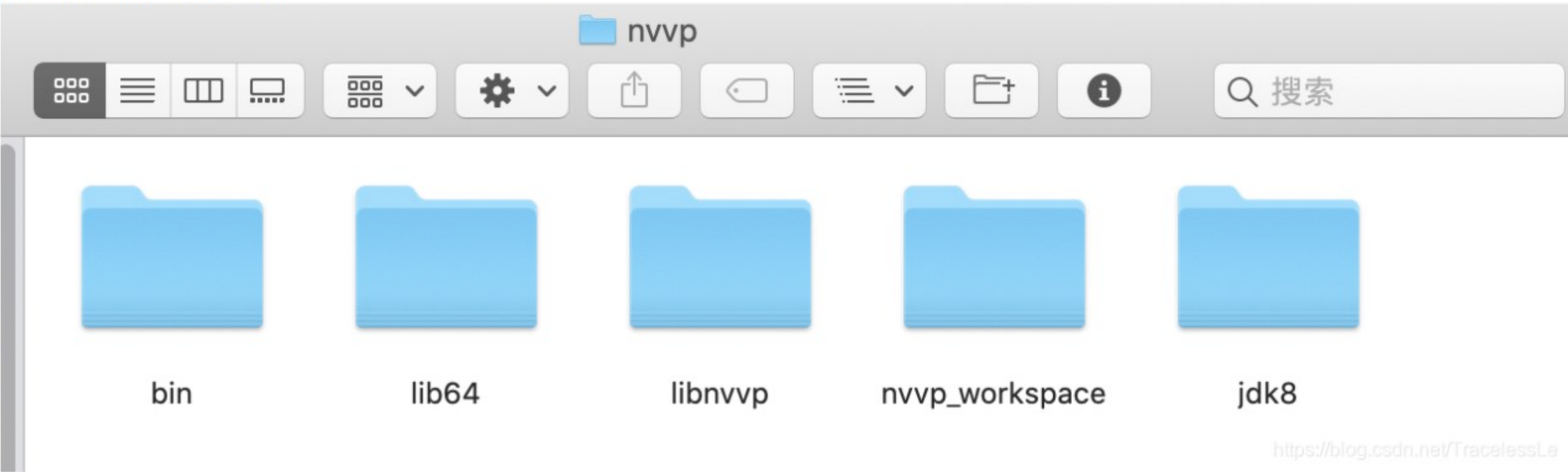
MacOS

Mac自从不再支持英伟达显卡后，只能使用nvvp工具查看profile文件。

要使Mac能查看nvvp日志，需要安装JDK环境和nvvp工具包。安装过程参考官网：<https://developer.nvidia.com/nvidia-cuda-toolkit-developer-tools-mac-hosts>

适用于macOS 系统版本11以下

1. 下载nvidia-visual-profiler-mac-11.1-28936279.dmg ([官网](#)或者[CSDN资源](#))
2. 下载zulu8.23.0.3-jdk8.0.144-macosx_x64.zip ([官网](#)或者[CSDN资源](#))
3. 解压放至某文件夹 (如/users/name/test/)



注：

- ①JDK可以不用安装，解压放在nvvp文件夹下即可。
- ②JDK如果不是上述版本，可能会出现无法使用的问题（弹出workspace选择界面后卡死）。

4. 使用

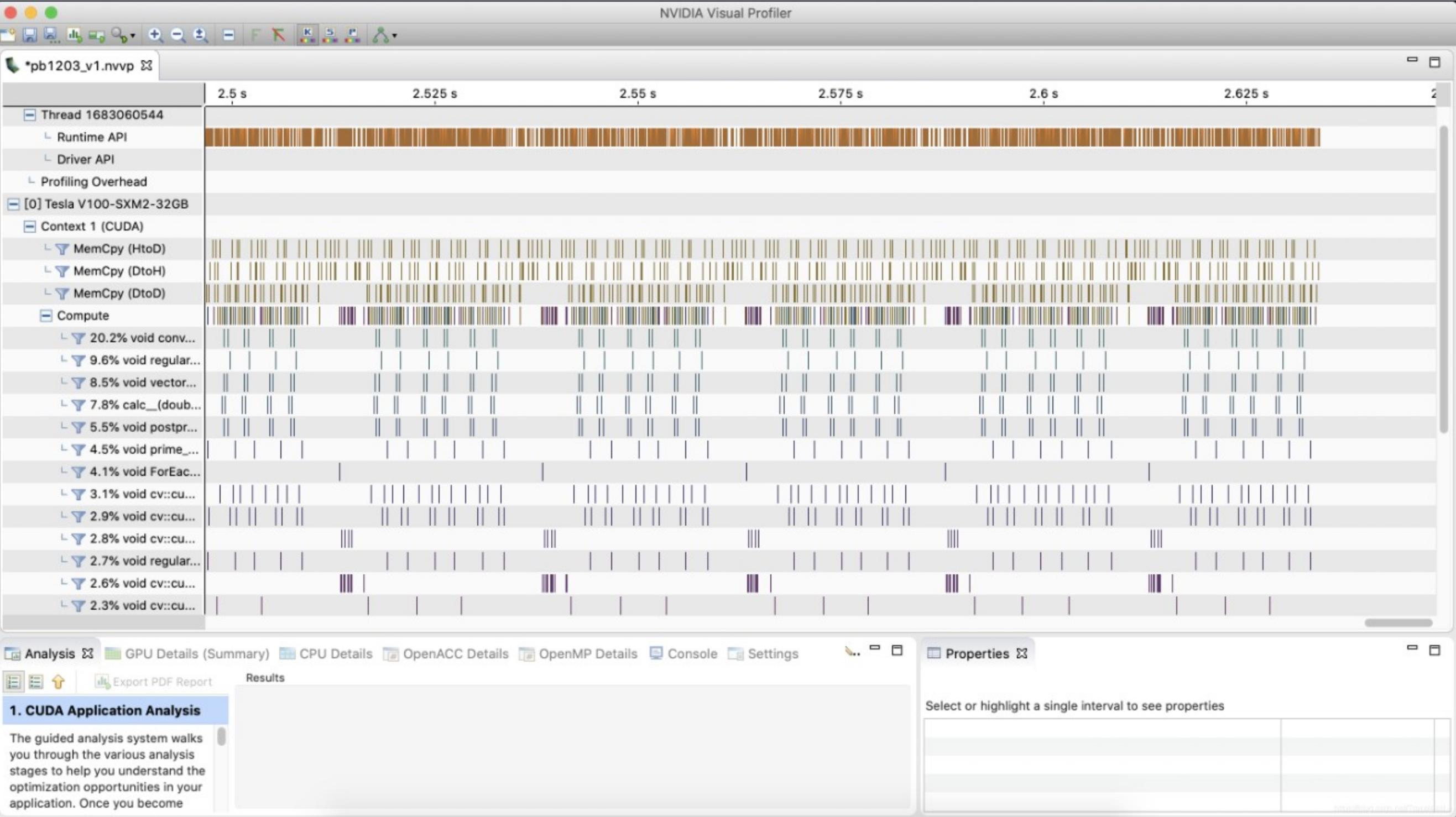
初次使用可能会被Mac系统拦截，这是由于安全性设置导致的。可以在下面的页面允许jdk相关程序运行（会调用jdk的bin目录下8个左右程序），也可以关闭该安全性设置（参考资料[6]）。



```
1 | cd nvvp
2 | ./bin/nvvp -vm /users/name/test/nvvp/jdk8/bin/java
```

需要注意-vm后的路径需为绝对路径。如不指定则自动寻找环境变量中的java。

Mac下打开nvvp文件显示如下，使用方式参考上述“打开nvvp日志文件”部分：



适用于macOS 系统版本11(Big Sur) 及以上

- 1. 下载nvidia-visual-profiler-mac-11.1-28936279.dmg ([官网](#)或者[CSDN资源](#))
- 2. 下载zulu8.23.0.3-jdk8.0.144-macosx_x64.dmg ([官网](#)或者[CSDN资源](#))
- 3. 点击zulu8.23.0.3-jdk8.0.144-macosx_x64.dmg安装jdk



4. 建立libjvm.dylib的软链接

```
1 | sudo ln -s /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home/jre/lib/server/libjvm.dylib /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home/lib/libserver.dylib
```

5. 使用

```
1 | cd nvvp
2 | ./bin/nvvp
```

需要注意在MacOS11及以上的版本无需再指定-vm参数。

初次使用可能会遇到 `'lib.dylib' cannot be allowed to run because its origin cannot be verified` 的报错，说明jdk运行时被Mac系统拦截，这是由于安全性设置导致的。打开系统安全性设置点击允许（‘System Preferences’ → ‘Security & Privacy’ → ‘General’）。

b. For macOS version 11 (Big Sur) and beyond

- Make sure that required java is in your JAVA_HOME. For example: `/Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home`
- Run nvvp executable.
`> ./nvvp`
- If you get the error when loading the JVM library, for example:
`JavaVM: Failed to load JVM: /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home/lib/libserver.dylib`
`JavaVM FATAL: Failed to load the jvm library.`
Then create symlink for libjvm.dylib and run nvvp again
`> sudo ln -s /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home/jre/lib/server/libjvm.dylib /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home/lib/libserver.dylib`
`> ./nvvp`
- If you get the error when attempting use the lib.dylib library, for example:
`'lib.dylib' cannot be allowed to run because its origin cannot be verified`
Then go to 'System Preferences' → 'Security & Privacy' → 'General' and click the 'Allow...' button for the blocked 'lib.dylib'.
Run nvvp again:
`> ./nvvp`

<https://blog.csdn.net/TracelessLe>



TracelessLe
码龄6年

欢迎关注
我的CSDN

参考资料

[1] [Nvidia Developer - Performance Analysis Tools](#)
[2] [NVIDIA Visual Profiler](#)
[3] [\[腾讯机智\] tensorflow profiling工具简介——nvprof和nvvp](#)
[4] [CUDA Toolkit v11.1.1 Doc - Profiler - Visual Profiler](#)
[5] [NVIDIA CUDA Toolkit - Developer Tools for macOS](#)
[6] [macOS Catalina\(10.15\)解决阻止程序运行“macOS无法验证此App不包含恶意软件”](#)
[7] [How to use NVIDIA profiler](#)
[8] [Understanding the Visualization of Overhead and Latency in NVIDIA Nsight Systems](#)
[9] [NVIDIA Nsight Systems](#)
[10] [Does calling a CUDA kernel multiple times affect execution speed?](#)



显示推荐内容