



# **Denglin Hamming™ V2**

## **Profiler 使用说明**

DL-DG/SW-043A-01

2023-7-30

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

## 商标和许可



和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

## 通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼栋5号1101室，江苏，中国

<http://www.denglin.ai>

email : support@denglin.ai

# 更新历史

版本	更新描述
01	第一次发布

# 内容目录

[更新历史](#)

[内容目录](#)

[概述](#)

[运行环境](#)

[环境变量设置](#)

[依赖环境安装](#)

[使用说明](#)

[dlprof](#)

[典型应用](#)

[参数说明](#)

[dlprof\\_viz](#)

[菜单视图](#)

[Timeline View](#)

[Host Timeline \(Threads\)](#)

[Hardware Queue Timeline \(Queues\)](#)

[Engine Timeline \(Engines\)](#)

[FAQs](#)

## 概述

Denglin Profiler 工具用于获取软硬件相关的 Profile 信息，包括当前运行的环境（Device, Context, Stream）信息、硬件提供的 Counter 和 Timestamp 信息、相关 Command（DMA, TU, CU等）信息、Host API Trace 信息等，并对信息完成初步的分析。

Profiler 提供以 Metric 为基础的分析结果，帮助使用者了解程序运行情况并进行上层应用优化。

Profiler 工具面向两种使用场景，Kernel Profile 和 Application Profile。

- Kernel Profile：用于单个 Kernel 的 Profile，提供针对当前 Kernel 相关性能 Counters，指导 Kernel 的编写。  
【TBD】

说明：【TBD】表示目前未完善，待后续补充。本文后续【TBD】均表示此意，不再单独解释。

- Application Profile：用于整个 Application 的 Profile，获取 Device 和 Host 执行的相关时间线（Timelines），分析相关调度算法、设备利用率、软件 Overhead 等。

在这里，对 Profile 的几个概念解释如下：

- **Pass**

由于硬件设置需要执行多次来获取所有的Counter，即每次执行为一个Pass，每个Pass通过pass\_index来进行区分。

- **Range**

在单个 Pass 的 Profile 过程中，需要区分不同的 Profile Range。对于 Application Profiling，Profile Range为程序开始到程序结束；对于 Kernel Profiling，Profile Range 为每个 Kernel 开始执行到结束执行。

- **Metric**

Metric 为 Profiler 根据相应的软硬件信息分析得到的性能信息，主要分为两类，分别为 Timestamp 和 Counter，其中 Counter 的意义取决于相关硬件的定义。

## 运行环境

Denglin Profiler 工具由两个应用程序组成，dlprof 和 dlprof\_viz。

其中，dlprof 用于 Profile 数据的收集和存储。dlprof 的运行依赖内部 SDK。

dlprof\_viz 用于 Profile 数据的可视化和分析，dlprof\_viz 的运行依赖于内部 SDK 和 OpenGL 环境。

## 环境变量设置

程序的运行依赖于内部 SDK，需要设置好 SDK 相应环境变量，命令如下：

```
source $SDK_ROOT/env.sh
```

## 依赖环境安装

dlprof\_viz 的运行依赖于 OpenGL，需要提前安装 OpenGL 运行环境。

在 Linux 系统中，以Ubuntu 20.04为例，命令如下：

```
# 桌面GL环境
sudo apt update && sudo apt install freeglut3-dev mesa-utils

# 远程环境
TBD

# 检查GL环境是否可用
glxgears # 能够正确执行无错误
```

## 使用说明

Denglin Profiler 工具由两个应用程序组成，dlprof 和 dlprof\_viz。

其中，dlprof 用于 Profile 数据的收集和存储。dlprof\_viz 用于 Profile 数据的可视化和分析。

下面对这两个程序的使用做一些说明和示例。

### dlprof

dlprof 提供两种运行模式，Kernel Profiling 和 Application Profiling。

- Application Profiling：由于分析整个应用的执行，执行顺序和未 Profiling 时相同，能够建立完整的执行 Timeline。
- Kernel Profiling：用于分析单个 Kernel 的执行 (tu or cu)，在 Profiling 时，保证每个 Kernel 被单独的 Profile，即每个 Kernel 可以获取相对独立的 Metrics，包括 Timestamps 和 Counters。【TBD】

需要注意的是：

1. Graph 模式下不支持 Kernel Profiling。
2. 当前不支持多个 Context 的 Profiling，当应用存在多个 Context 时，结果可能会不准确。【TBD】

### 典型应用

#### • Application Profiling

仅获取 Device 所有 Timelines，最小化 Profiling 的 Overhead：

```
dlprof --save test.dlprof --quiet cmd [arg]...
```

#### • Kernel Profiling 【TBD】

1. 以 Kernel (tu, cu) 为 Profile Range 获取指定 pass 的 Metrics。

```
dlprof --include .* --exclude "" --save test.dlprof --quiet --pass 0 cmd [arg]...
```

2. 以 Kernel (tu, cu) 为 Profile Range 获取指定 pass 的 Metrics，并导出为 JSON 格式方便外部工具分析。

```
dlprof --include .* --exclude "" --save test.dlprof --quiet --pass 0 cmd [arg]...
```

3. 以 Kernel (tu, cu) 为 Profile Range 获取指定所有 pass 的 Metrics ( 自动进行 Application Replay ) , 并导出为 JSON 格式方便外部工具分析。

```
# pass [0 - 8(inclusive)]
dlprof --include .* --exclude "" --save test.dlprof --quiet --pass 0-8 cmd
[arg]...
```

参数说明

参数说明如下表所示。

参数	类型	默认值	应用场景	应用范围	说明
--device	int	0	Kerenl, App	-	用于指定需要 Profile 的设备序号。
--include --exclude	Regex string	.timestamp /su.	Kernel	TBD	在 Kernel Profiling 时, 通过正则表达式指定所有需要包含和排除的 Metrics, 可能包含多个 Group (Pass) 的多个 hardware counter, 需要通过 --pass 来选择实际采集的 counter。
--list	SWITCH	OFF	Kernel	TBD	列出被选中的 Metrics 名称, 用于判断 --include, --exclude 是否设置正确。
--pass	int or range	0	Kernel	TBD	可以为单个 pass ( 比如0 ), 或者 range ( 如2-4 ), 当参数为 range 时, dlprof 会自动进行 Application Replay, 以获取所有指定 pass 的 counters。
--host_func_mask	int(bit mask)	0	App	TBD	-
--save	filename(*.dlprof)	""	Kernel, App	-	指定原始数据存储的文件名, 设置为空表示不存储。
--mode	enum{app, kernel}	app	Kernel, App	-	用于指定 dlprof 的工作模式。

参数	类型	默认值	应用场景	应用范围	说明
--quiet	SWITCH	OFF	Kernel, App	-	用于关闭额外信息的打印，建议设为ON，否则会有大量信息被打印。
--help	SWITCH	OFF	Kernel, App	-	打印帮助信息并退出。
cmd [args...]	Command line	NA	Kernel, App	-	需要 Profile 的应用及其参数。

dlprof\_viz

dlprof\_viz 用于可视化 Application Profiling 的结果，使用方法如下：

```
dlprof_viz examples/test.dlprof
```

菜单视图

dlprof\_viz 的菜单简单介绍如下：

- 1. Device Info - 查看 Device Info。
- 2. Stream Info - 查看 Stream Id 与 Queue Id 的映射关系。

说明：当 Stream 与 Queue Id 不是一一对应时，仅选择其中一个进行显示。
- 3. Block Stats - 对 Device Command 的执行时间进行统计分析。
- 4. Highlight Graph - 对指定 Graph Id (gid) 进行高亮显示。
- 5. Highlight Block - 通过正则表达式对 Command 进行搜索并高亮。
- 6. Highlight Reset - 重置所有高亮 Command。
- 7. Set Block Visible - 过滤需要显示的 Command 类型，默认只显示 Async Command。

Timeline View

dlprof\_viz 将 Timeline 的显示分为三个部分:

- Host Timeline (Threads)：用于显示软件不同 Module 的 API 调用顺序和相关参数。
- Hardware Queue Timeline (Queues)：用于显示不同 Hardware Queue 上 Command 执行的顺序和相关参数。
- Engine Timeline (Engines)：以 Hardware Engine 为单位，显示相关 Command 的执行顺序和相关参数。

下表对 Timeline 快捷键进行说明。

快捷键	功能	说明
滚轮	缩放	-



快捷键	功能	说明
右键区域选择	区域放大	需要单击 Timeline 的空白处

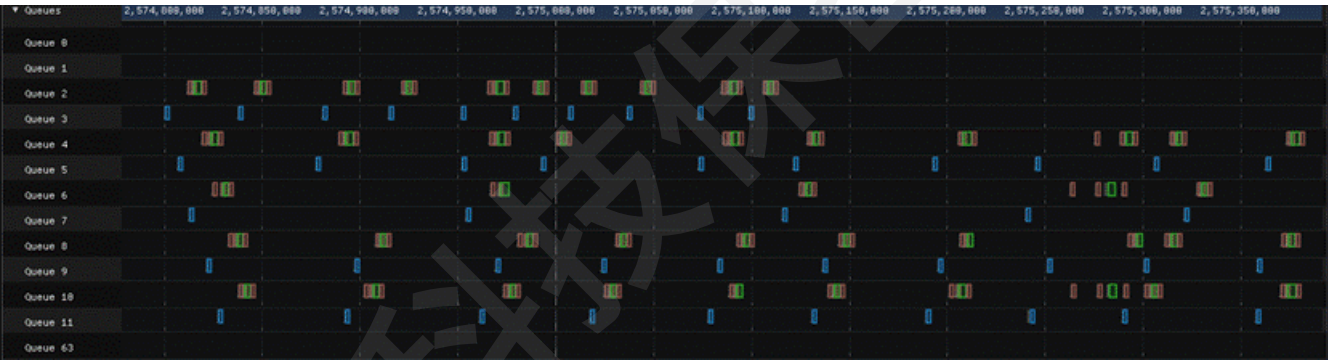
Host Timeline (Threads)

Host Timeline 以每个 Host Thread 为单位，将单个 Host Thread 上不同 Module 的 API 调用，以 Timeline 的形式进行可视化。

Hardware Queue Timeline (Queues)

以实际 Hardware Queue 的形式可视化相关 Command 的执行，Queue 的总数为64，其中 Queue 63 为内部 Queue，支持的 Command 如下表，Command Trace 结果一般由四部分组成：

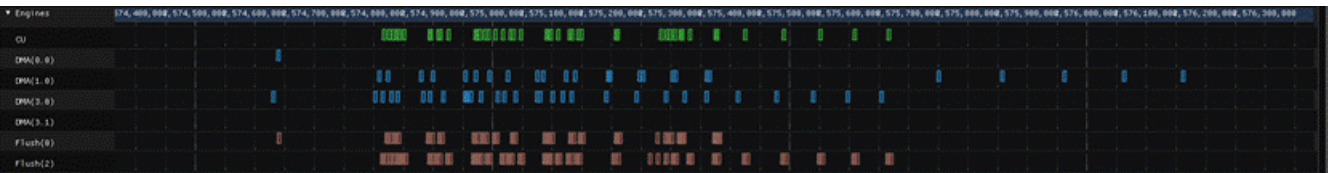
- Args [optional] - 相关API调用的参数，包括传入参数和内部参数(前后带双下划线，如 `__stream__`)
- Metric [optional] - Command 执行相关的 Timestamp。
  - time\_start, time\_end，time\_elapsed：开始、结束、和实际执行的时间。
  - issue\_overhead：从下发命令到实际执行的间隔。
  - `__repr__`：Command 的名称，Graph Mode 可由上层用户指定（node\_name）



Command	Label	Type	API
DMA	DeviceToHost HostToDevice DeviceToDevice	Async	Memcpy/Memcpy4D
Cu	Kernel Name	Async	LaunchCu
Tu	<Tu>	Async	LaunchTu

Engine Timeline (Engines)

以实际 Hardware Engine 的形式可视化相关 Command 的执行，可以通过此视图来分析相关 Hardware Engine 使用效率，其中 Command 信息与 Hardware Queue Timeline (Queues) 相同。



## FAQs

Q：如何获取 Command 或 API 执行的时间？

A：Profiling 中获取的 timestamp 的单位为 cycles，需要将 cycles 除以时钟频率得到时间。

时钟频率获取方法如下：dlprof\_viz - 菜单：View -> Device Info

登临科技保密材料