



华为技术有限公司	部门名称	密级
	Hyper Stack	内部公开
	项目名称	页数
	ARM64 性能分析工具开发	共 39 页

Malluma V2.0

Performance Analysis User's Guide

Prepared by		Date	
拟制		日期	
Reviewed by		Date	
评审人		日期	
Approved by		Date	
批准		日期	
Authorized by		Date	
签发		日期	



Huawei Technologies Co., Ltd.

华为技术有限公司

All rights reserved
版权所有 侵权必究



Revision Record 修订记录

Date 日期	Revision Version 修订版本	Sec No. 修改章节	Change Description 修改描述	Author 作者
2018-05-15	V1.0	All	1.完成大纲 2.1616 环境支持 7 种采集分析	
2018-10-10	V2.0	2、5.1.8、 5.4.15、 5.1.8、 5.4.1、 5.4.12、 5.4.15、7、 7.1、7.2	1.新增系统登录账号说明 2.1620 环境新增支持所有 1616 环境采集分析 3.1620 环境新增支持 False Sharing 采集分析 4. 仅针对 Hisilicon CPU Hisilicon 1620 环境 ge 和 Id 采集分析支持勾选 Profiling LLC and DDR 之后可选性设置 Sampling interval(ms)值 5. sch 采集新增支持 Time Consumption 分析 6.新增 Malluma 使用 FAQ 说明	



目录

Revision Record 修订记录	2
1 分析系统启动	5
2 用户管理	5
3 工程管理	6
4 分析选择	8
4.1 新建分析	8
4.2 导入分析结果	8
5 开始分析	8
5.1 分析类型选择	8
5.1.1 General Exploration 分析	9
5.1.2 Scheduling 分析	9
5.1.3 Disk IO 分析	9
5.1.4 Locks and Waits 分析	10
5.1.5 LLC&DDR 分析	10
5.1.6 Java Mixed-Mode 分析	10
5.1.7 Network Input and Output 分析	11
5.1.8 False Sharing 分析	12
5.2 分析对象选择及参数配置	12
5.2.1 Launch Application	12
5.2.2 Attach to Process	13
5.2.3 Profile System	14
5.2.4 命令行脚本展示	14
5.3 分析启动及状态反馈	15
5.3.1 启动分析	15
5.3.2 状态反馈	16
5.4 分析结果查看	16
5.4.1 Summary 分析	16
5.4.2 Function 分析	19
5.4.3 Timechart 分析	21
5.4.4 Zoomview 分析	22
5.4.5 Code 分析	23
5.4.6 Resource 分析	25
5.4.7 Control Flow 分析	26
5.4.8 Latency 分析	27
5.4.9 Disk IO 分析	28
5.4.10 Locks and Waits 分析	28
5.4.11 Locks and Waits Code 分析	29
5.4.12 LLC&DDR 分析	30
5.4.13 Flame Graph 分析	32
5.4.14 Network Input and Output 分析	32
5.4.15 False Sharing 分析	33
6 命令行界面支持	34



6.1	功能介绍	34
6.2	命令参数	36
6.3	示例	37
6.3.1	服务器数据采集	37
7	FAQ	38
7.1	Websocket 服务中断	38
7.2	Malluma 采集周期	38
7.3	Malluma 部署成功但不能访问 Web 端	39

1 分析系统启动

完成 Malluma 分析系统安装、配置之后（参考 Install_Guide_Malluma_linux.docx），在 Web 地址栏输入 Server IP 地址启动分析系统。用户登陆后（如图 1.1），自动加载 Sever 中的已有工程及结果到工程导航栏 Project Navigator（如图 1.2）。



图 1.1

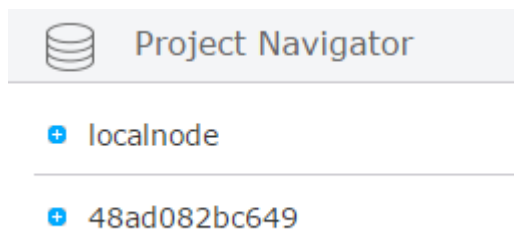


图 1.2

2 用户管理

管理员账号（admin/Admin12#\$）登陆 Malluma 系统平台，支持用户管理，点击“User management”（如图 2.1），弹出对话框添加/删除/更新用户（如图 2.2），可以添加管理员用户或普通用户：

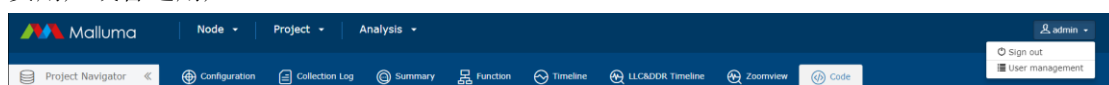


图 2.1

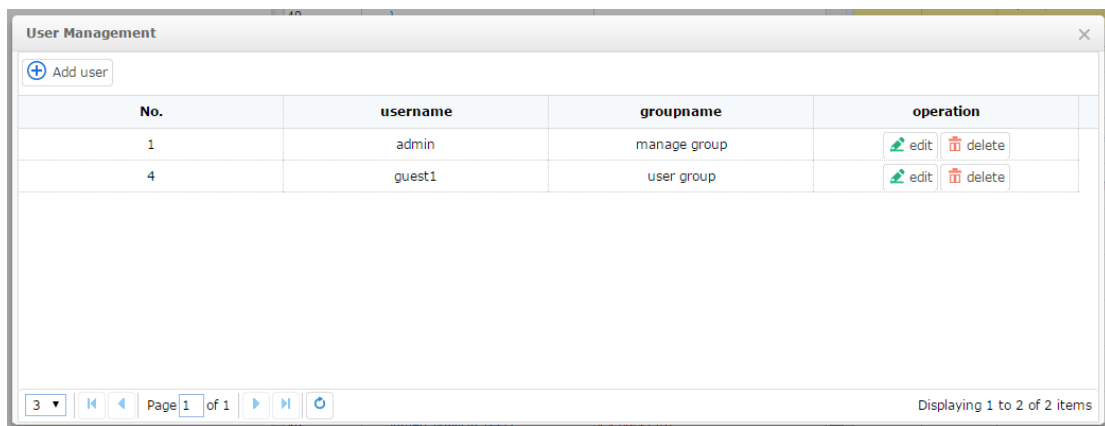


图 2.2

权限说明:

- (1) **admin** 管理员为超级管理员,可以对所有用户的节点、工程、采集分析有浏览、修改、删除权限。
- (2) **guest** 用户,只支持浏览,无法新建分析。
- (3) **user** 用户(普通用户),支持新建节点、工程、采集分析,且可对自建的节点、工程、采集分析进行修改、删除操作;对 **admin** 管理员新建的节点、工程、采集分析,只支持浏览不支持其他修改、删除操作;不同普通用户之间只有浏览权限,无其他操作权限(修改或删除节点、工程、采集分析权限)。

3 工程管理

创建工程,选择 **Project** 菜单中 **New Project** (如图 3.1),弹出对话框输入工程名(如图 3.2)。

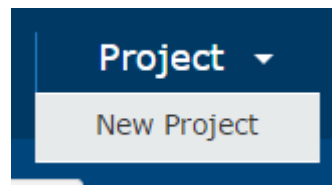


图 3.1

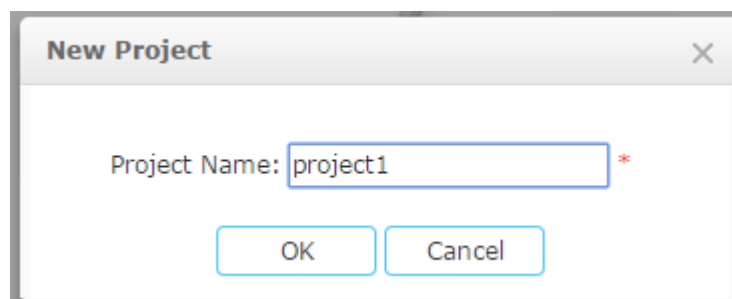


图 3.2

双击选中工程,展开工程下的采集结果,此时可以新建分析。双击已有的采集分析结果,加载分析结果到 **Summary** 等分析界面(如图 3.3)。

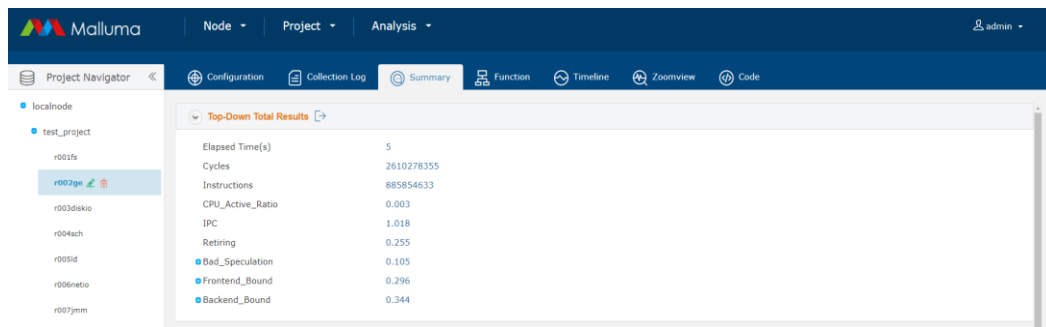


图 3.3

可重命名或删除工程名和结果名（如图 3.4），重命名直接在结果名称上进行编辑（如图 3.5），删除弹出确认对话框（如图 3.6）。

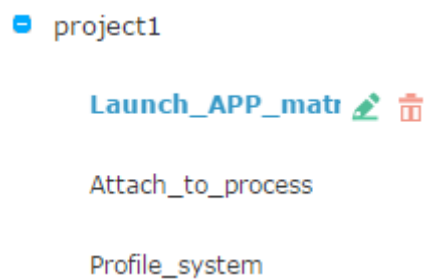


图 3.4

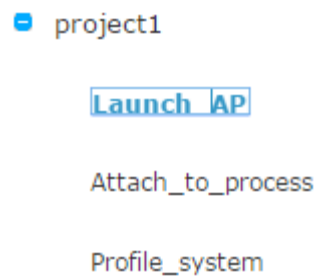


图 3.5

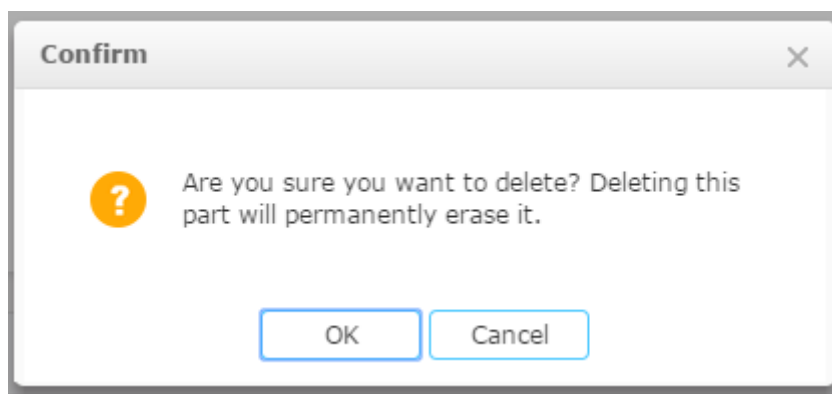


图 3.6

4 分析选择

4.1 新建分析

新建分析结果，选择 **Analysis** 菜单中 **New Analysis**（如图 4.1），显示可选择的分析类型和对象选择页面，可进行相应的设置。注意：先选中任一工程名。

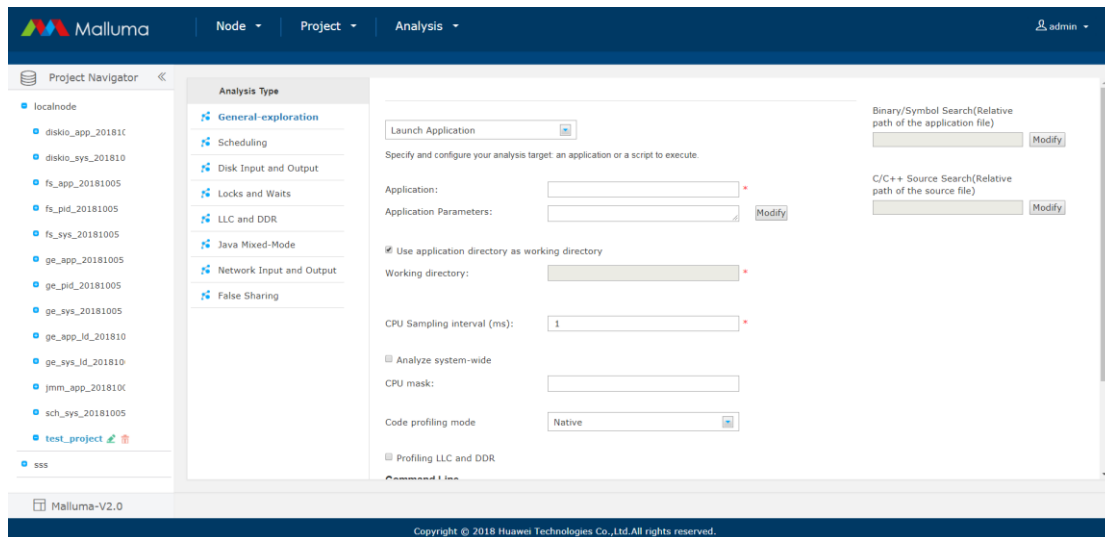


图 4.1

4.2 导入分析结果

导入分析结果，选择 **Analysis** 菜单中 **Import Results** 弹出导入对话框（如图 4.2），从后台选择已有的采集结果文件夹输入到 **Path** 中，执行导入时会加载采集分析到指定的工程目录下。

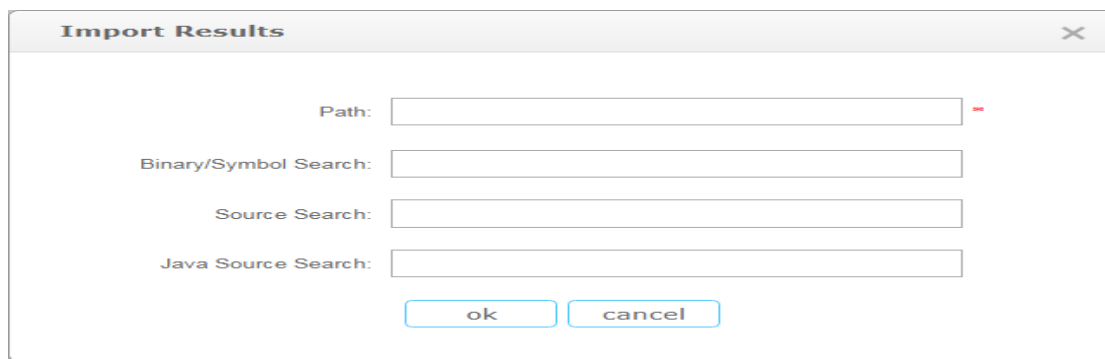


图 4.2

5 开始分析

5.1 分析类型选择

选择分析类型，从哪些维度来分析应用的性能瓶颈，比如微架构级 top-down 模型、算法级锁与等待、平台级 Disk I/O 分析等。

5.1.1 General Exploration 分析

General Exploration 分析,即微架构 top-down 模型分析方法,详见 Tuning Methodology 介绍。可选中分析方法 General-exploration, 如图 5.1.1。

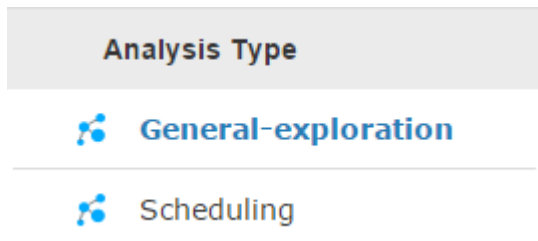


图 5.1.1

5.1.2 Scheduling 分析

Scheduling 分析,即调度轨迹分析方法,可选中分析方法 Scheduling, 如图 5.1.2。

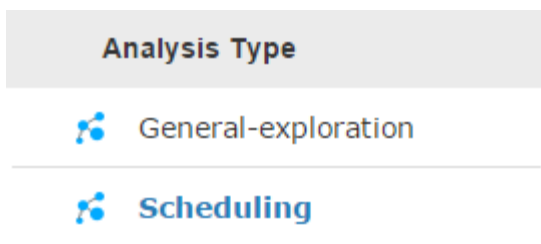


图 5.1.2

5.1.3 Disk IO 分析

Disk IO 分析,即磁盘输入输出分析方法,可选中分析方法 Disk Input and Output, 如图 5.1.3。

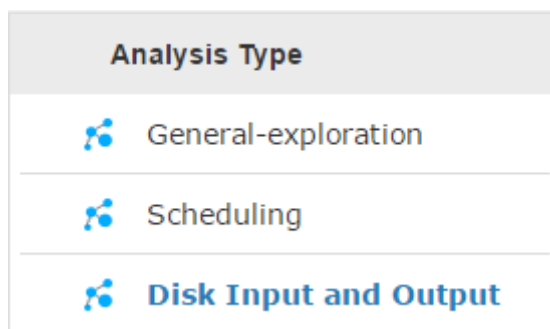


图 5.1.3

5.1.4 Locks and Waits 分析

Locks and Waits 分析,即 glibc 锁和等待分析方法,包括 sleep、usleep、mutex、cond、spinlock、rwlock、semaphore 接口调用引起的并发性能分析,可选中分析方法 Locks and Waits,如图 5.1.4。

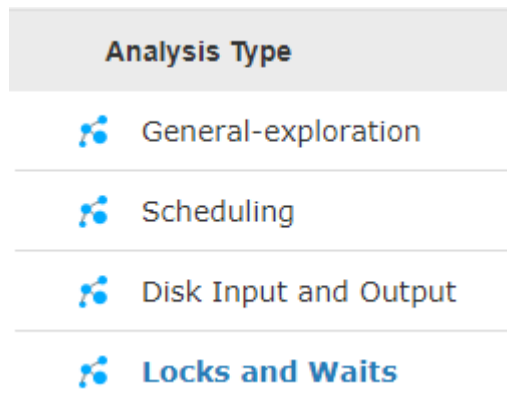


图 5.1.4

5.1.5 LLC&DDR 分析

LLC&DDR 分析,即分析 L3 Cache 和 DDR 内存的缓存命中率及读写的带宽信息,可选中分析方法 LLC and DDR,如图 5.1.5。

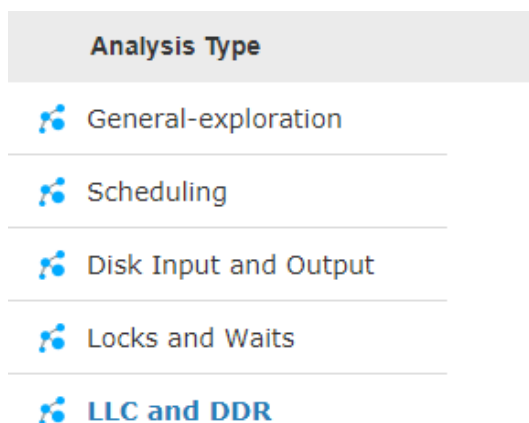


图 5.1.5

5.1.6 Java Mixed-Mode 分析

Java Mixed-Mode 分析,即支持分析 java 程序的代码,但称之为 Mixed-Mode,是因为不仅仅能采集到 java 方法,同时还能采集到 native 代码。可选中分析方法 Java Mixed-Mode,如图 5.1.6。

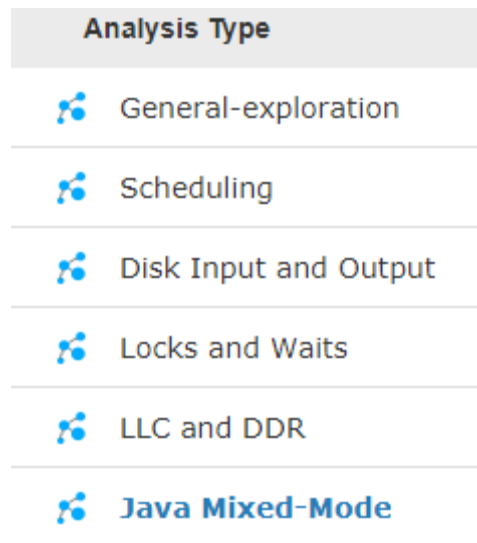


图 5.1.6

5.1.7 Network Input and Output 分析

Network Input and Output 分析,分析每个网口上的收发包统计信息,包括收发包带宽、收发包个数、收发包错包个数和收发包丢包个数。可选中分析方法 **Network Input and Output**,如图 5.1.7。

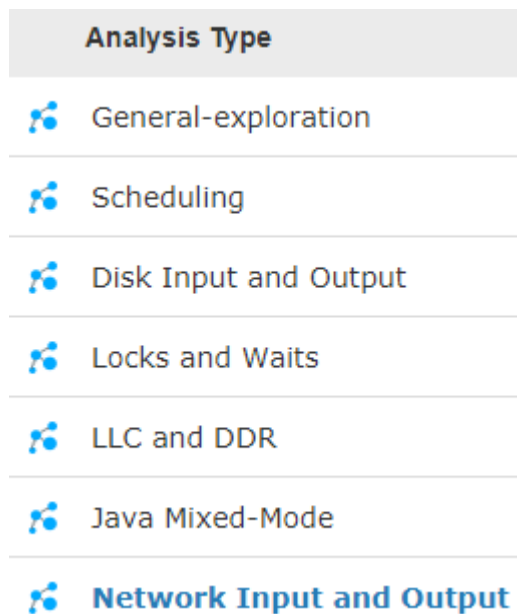


图 5.1.7

5.1.8 False Sharing 分析

False Sharing 分析，多进程数据共享分析（The analysis of data sharing for multi-threads），该采集分析仅支持 1620 环境。可选中分析方法 False Sharing，如图 5.1.8。

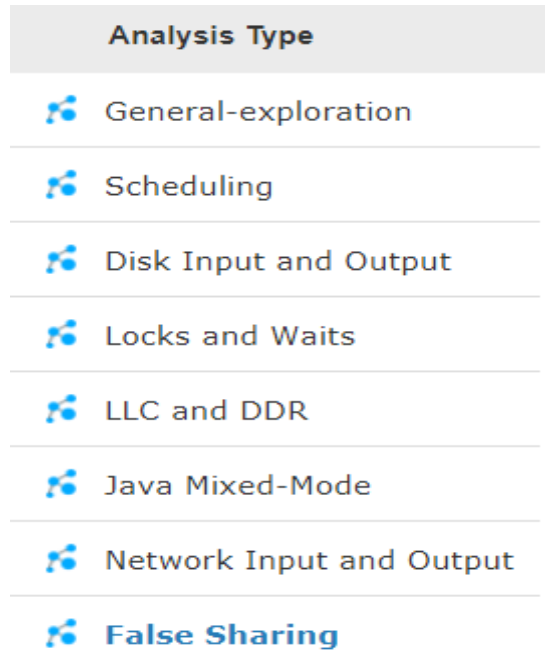


图 5.1.8

5.2 分析对象选择及参数配置

主要有三种形式的分析对象，分别为 Launch Application、Attach to Process、Profile system。

5.2.1 Launch Application

Launch Application 即采集启动的时候同时启动 Application（如图 5.2.1），采集时长受 Application 的执行时间来控制，适用于 Application 运行时间较短的场景。

参数配置：

Application 参数，配置被采集 Application 的绝对路径。

Application Parameters 参数，配置 Application 本身的执行参数。提供 Modify 功能。

Working directory 参数，配置 Application 的运行绝对路径，默认与 Application 的绝对路径一致。

CPU Sampling interval 参数，配置数据采样时间间隔，默认值 1。阈值范围[1,1000]。

Analysis system-wide 参数，勾选之后实现全系统采集。

CPU mask 参数，配置需要关注的 CPU 核。

Modify 功能，即弹出较大的文本编辑框。

Profiling LLC and DDR：是否采集 LLC and DDR 数据，若需要采集，勾选该选项；若是 Hisilicon CPU Hisilicon 1620 勾选 Profiling LLC and DDR 后同时支持 Sampling

interval(ms) (Sampling interval(ms) 支持 10/50/100/1000 四种选择) (勾选该选项前需要进行环境预置, 具体操作和要求见 5.4.12 章节)。

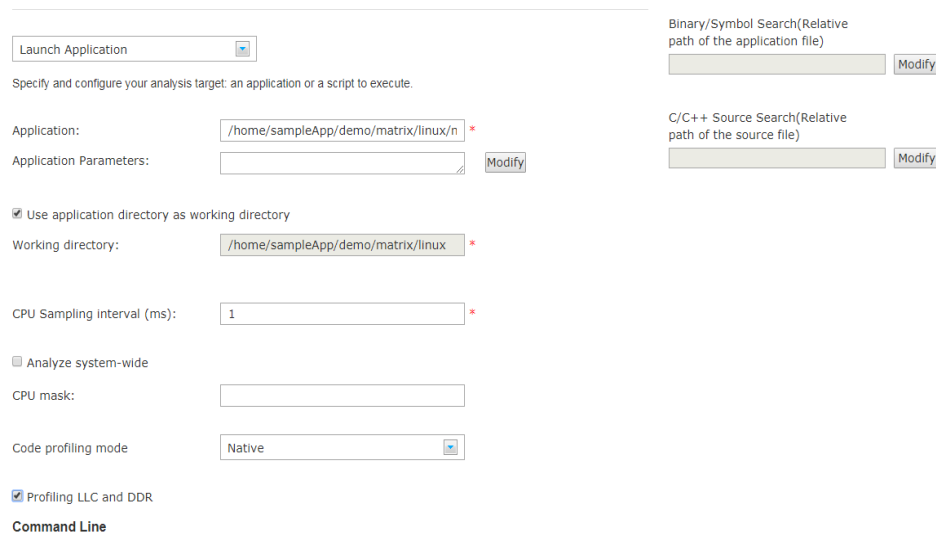


图 5.2.1

5.2.2 Attach to Process

Attach to Process 即采集启动的时候 Server 中应用正在运行 (如图 5.2.2), 采集时长需要配置参数控制, 适用于某些应用需要长时间持续运行的场景。

参数配置:

PID 参数, 配置被采集应用的 PID 号。

Collection duration 参数, 配置采集时长。

CPU Sampling interval 参数, 配置数据采样时间间隔, 默认值 1。阈值范围[1,1000]。

Profiling LLC and DDR: 是否采集 LLC and DDR 数据, 若需要采集, 勾选该选项; 若是 Hisilicon CPU Hisilicon 1620 勾选 Profiling LLC and DDR 后同时支持 Sampling interval(ms) (Sampling interval(ms) 支持 10/50/100/1000 四种选择) (勾选该选项前需要进行环境预置, 具体操作和要求见 5.4.12 章节)。



图 5.2.2

5.2.3 Profile System

Profile System 即采集整个 **Server OS** 系统（如图 5.2.3），无需关注系统中有哪些类型的应用在运行，采集时长需要配置参数控制，适用于多业务混合运行和有 **Child Process** 的场景。

参数配置：

Collection duration 参数，配置采集时长。

CPU Sampling interval 参数，配置数据采样时间间隔，默认值 1。阈值范围[1,1000]。

CPU mask 参数，配置需要关注的 CPU 核。

Profiling LLC and DDR：是否采集 LLC and DDR 数据，若需要采集，勾选该选项；若是 Hisilicon CPU Hisilicon 1620 勾选 **Profiling LLC and DDR** 后同时支持 **Sampling interval(ms)**（**Sampling interval(ms)** 支持 10/50/100/1000 四种选择）（勾选该选项前需要进行环境预置，具体操作和要求见 5.4.12 章节）。

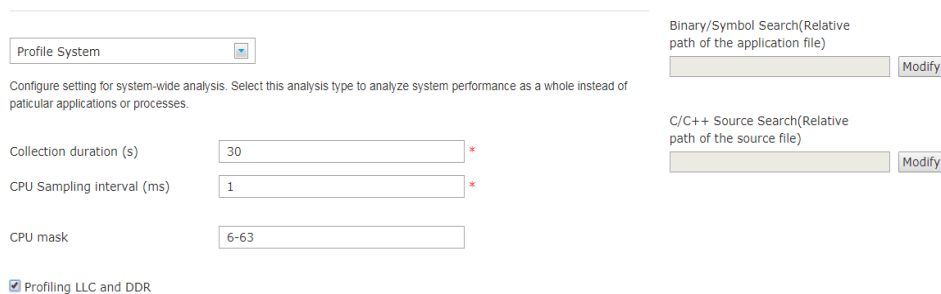


图 5.2.3

5.2.4 命令行脚本展示

根据三种采集方式 **Launch Application**、**Attach to Process**、**Profile System** 的配置参数自动关联生成命令行采集脚本（如图 5.2.4），可以直接复制到命令行执行，支持新建采集和采集结果查看两个功能场景。

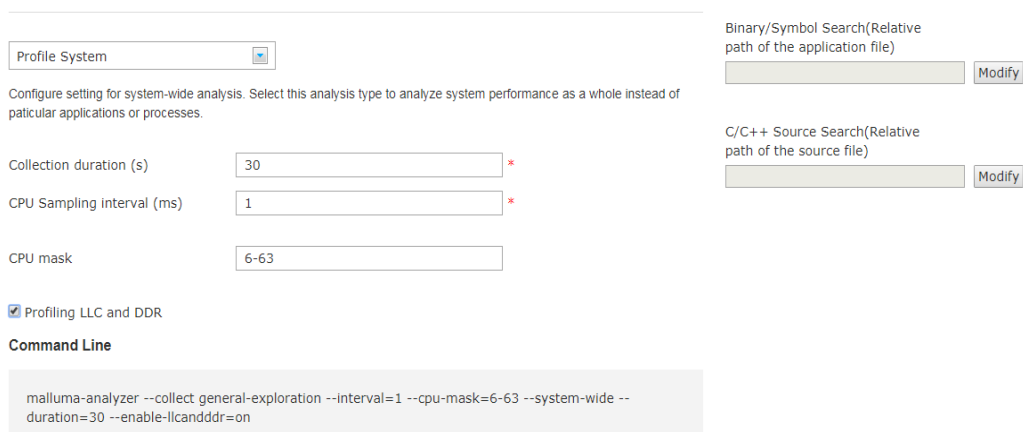


图 5.2.4

5.3 分析启动及状态反馈

5.3.1 启动分析

完成分析对象和参数配置之后，点击 **Start** 按钮启动采集（如图 5.3.1.1），执行采集过程进入到 **Collection Log** 界面。点击 **Cancel** 按钮取消采集。



图 5.3.1.1

启动采集之前或采集之后，可以点击 **Binary/Symbol Search** 的 **Modify** 按钮（如图 5.3.1.2），弹出对话框配置带符号表的 **Application** 的路径或符号表的路径。配置符号表信息，适用于 **Application** 运行时不含符号表信息的场景。配置方式如下：

1) 用户配置的相对路径下面可以放带 **debug** 信息的二进制文件。例如，采集时运行的二进制应用文件所在路径为 `/home/prog1`，用户在弹出的对话框配置相对路径为 `/symbol`，则用户需要将带 **debug** 信息的 `prog1` 二进制文件放在服务器的 `/symbol/home/prog1` 路径。

2) 用户配置的相对路径下面可以放不带 **debug** 信息的二进制文件和与之对应的 **debuginfo** 文件。例如，采集时运行的二进制文件所在路径为 `/home/prog1`，用户配置相对路径为 `/symbol`，则用户需要将不带 **debug** 信息的 `prog1` 二进制文件放在服务器的 `/symbol/home/prog1` 路径，将 **debuginfo** 文件放在服务器的 `/symbol/home/debuginfo` 路径。

3) 用户配置符号表相对路径后，代码映射时，所有函数的代码映射都会在配置的路径下进行查找，在采集全系统的情况下，会涉及采集众多的二进制程序，每个二进制程序的函数如需要代码映射，都要在相对路径下进行 1) 或 3) 步描述的操作。

启动采集之前或采集之后，可以点击 **Source Search** 的 **Modify** 按钮（如图 5.3.1.2），弹出对话框配置 **APP** 的源码路径。配置源码路径信息，以便查看源代码与 **PMU** 指标的映射关系。配置方式如下：

用户在配置的相对路径下面放置 **C** 源代码文件。例如，采集的应用程序编译时源文件所在路径为 `/home/prog1.c`，用户在弹出的对话框配置相对路径为 `/Source`，则用户需要将 `prog1.c` 文件放在服务器的 `/Source/home/prog1.c` 路径。

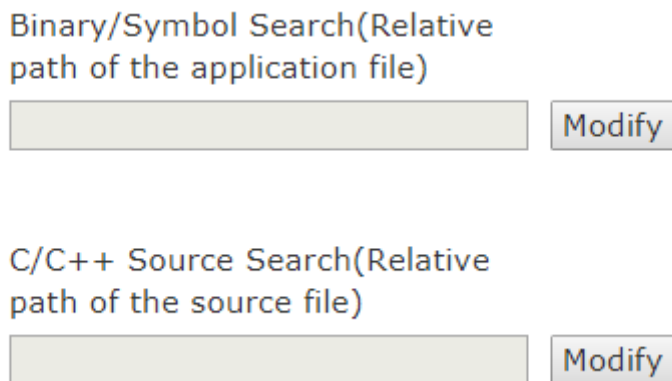


图 5.3.1.2

如果新建的分析 **Java Mixed-Mode** 类型或者在 **General Exploration** 分析类型里面选择 **Code profiling mode** 为 **Java Mixed-Mode**, 则会显示 **Java Source Search** 对话框 (如图 5.3.1.3), 这个时候要把需要映射源码的 .java 文件放到该目录下才能支持代码映射功能:

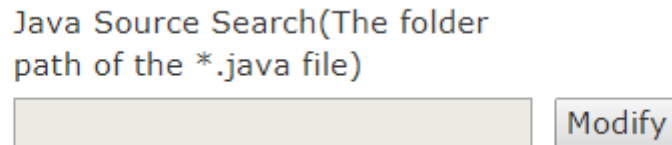


图 5.3.1.3

5.3.2 状态反馈

基于 **Collection Log** 功能, 直接打印采集日志信息到界面 (如图 5.3.2), 反馈采集成功的状态和解析的日志信息, 以及采集失败或解析失败的报错信息。支持新建采集分析和采集结果查看两种功能场景。

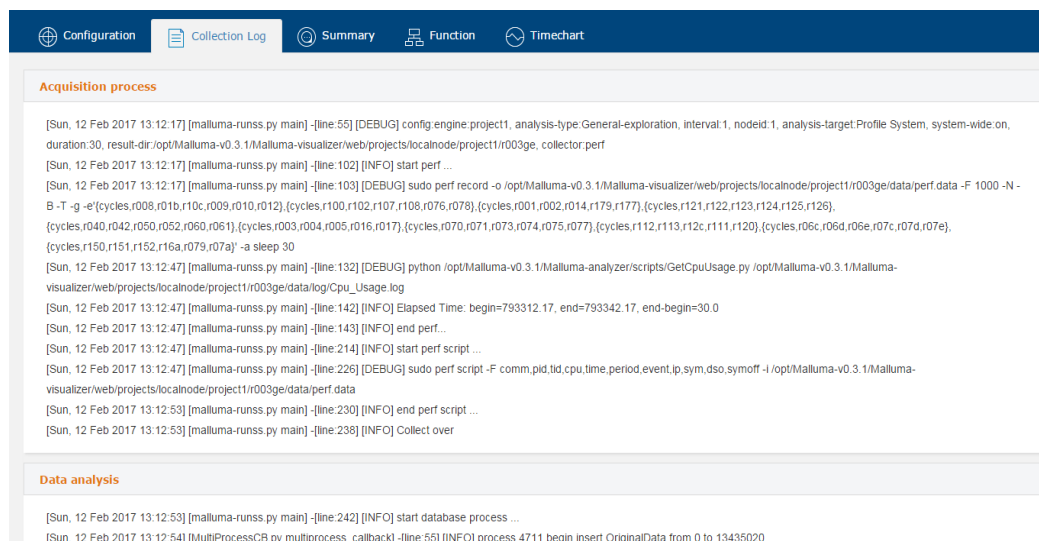


图 5.3.2

5.4 分析结果查看

5.4.1 Summary 分析

采集解析完成之后, 生成 **Summary** 分析报告。

➤ **Top-down 整体分析** (如图 5.4.1.1)

提供 **Top-down** 模型的整体值并标识主要性能瓶颈及优化建议。

Top-Down Total Results	
Elapsed Time(s)	33.770
Cycles	490766058
Instructions	31913673
CPU_Active_Ratio	0.000
IPC	0.650
Retiring	0.217
Bad_Speculation	0.088
Frontend_Bound	0.246
Backend_Bound	0.450

图 5.4.1.1 对应 General-exploration 分析

➤ Statistics 整体分析（如图 5.4.1.2）

Statistics	
Elapsed Time(s)	41.350
Instructions	93152904993
Cycles	1293646904206
CPI Rate	13.887
Event Types	
sched:sched_switch	10089
sched:sched_wakeup	4965

图 5.4.1.2 对应 Scheduling 分析

➤ Time Consumption 整体分析（如图 5.4.1.3）

Task	Time(ms)	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4
watchdog/19:144	Wait-Blocked					
sysmonit-47788	Running					
watchdog/75:530	Running					
sysmonit-47783	Wait-Blocked					
sysmonit-47782	Running					
kworker/12:2-mm:6201	Running					
sshd:5486	Wait-Blocked		0.894601999957			
check_sshd.sh:47741	Running		9.20000020415e-05			
check_sshd.sh:47741	Wait-Blocked					
kworker/20:2-mm:6204	Running					
kworker/u394:1-78948	Running					
sshd:47799	Wait-Blocked					
mysqld:30291	Running					
mysqld:30291	Wait-Blocked					
awc:47799	Wait-Blocked					0.00202200002968

图 5.4.1.3 对应 Scheduling 分析

➤ Disk IO 整体时延统计分析（如图 5.4.1.4）

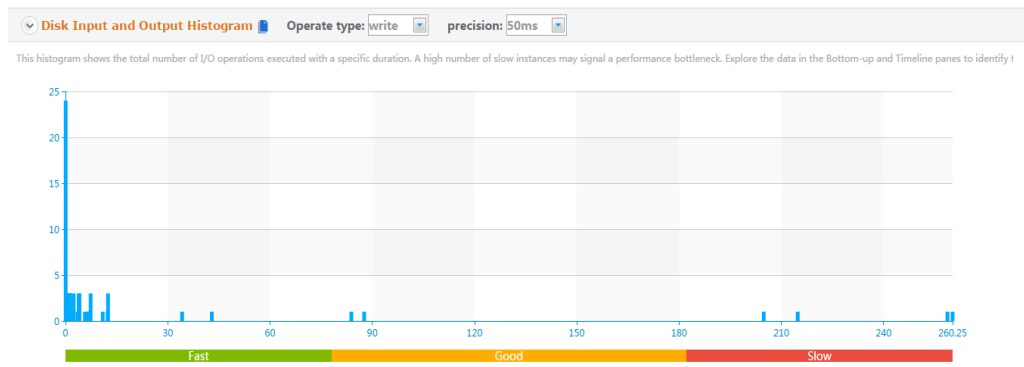


图 5.4.1.4 对应 Disk IO 分析

- Locks and Waits 统计分析（如图 5.4.1.5）（注：Wait Time 时间是所有线程等待时间的总和，在多线程情况下可能会大于程序执行时间）

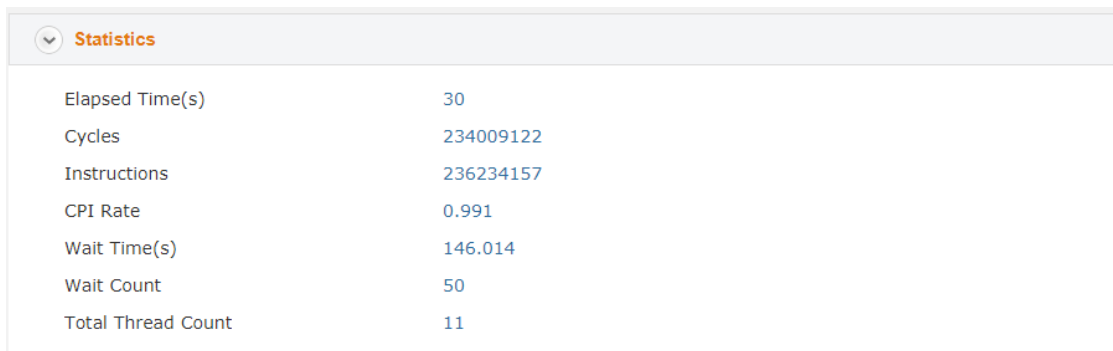


图 5.4.1.5 对应 lw 分析

- 采集 OS 系统基本指标

- CPU Usage（如图 5.4.2）

提供采集过程中 CPU Usage 的时序数据，以及平均值。

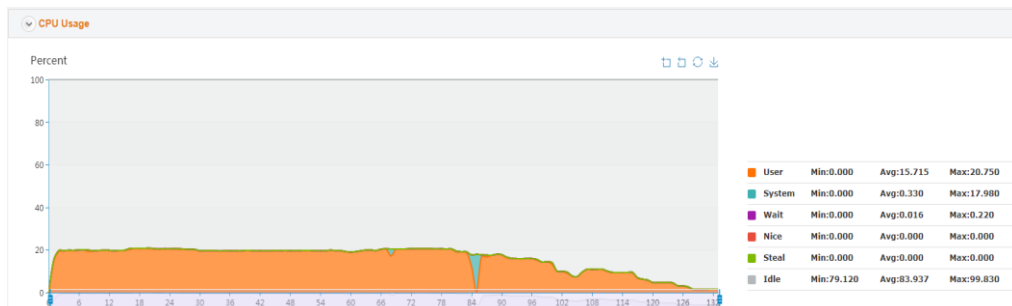


图 5.4.2

- 采集平台基本信息（如图 5.4.3）

提供采集执行 OS 系统的 CPU、OS、kernel 等环境信息，以及采集结果时间、结果大小等基本信息。

Collection and Platform Info	
Operating System:	4.1.36-vhulk3.8.1.B800.aarch64 Linux
Computer Name:	server-for-toolsdevelope
Result Size:	1338.258MB
Collection start time:	2017-02-12 13:06:28
Collection end time:	2017-02-12 13:06:58

图 5.4.3

5.4.2 Function 分析

- 查看 Function 的 top-down 指标值（如图 5.4.4-5.4.7）

分析不同函数对应 top-down 模型的各指标值，按选定指标排序查看其 TopN 热点函数，可知针对某些 PMU 指标哪些函数是主要瓶颈。（注意：cycles 太小的函数，对应其他指标值可行度低。）

Configuration Collection Log Summary Function Timeline								
Grouping: Function/Callstack		Flame Graph						
	Function/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTBBuBTB_Rate	PC_Write
1	total	1745109198135	8985245010	0.051	0.017	0.005	0.003	0.007
2	lib_mbedtls	1742915649938 (99.874%)	8702552279 (96.862%)	0.050	0.017	0.004	0.003	0.007
3	lib_mbedtls	790399779 (0.045%)	43689254 (0.486%)	0.553	0.184	0.024	0.016	0.136
4	lib_mbedtls	178236470 (0.01%)	35964448 (0.4%)	2.018	0.673	0.003	0.028	0.026
5	lib_mbedtls	139999995 (0.008%)	6627369 (0.074%)	0.473	0.159	0.026	0.033	0.118
6	lib_mbedtls	93107343 (0.005%)	3580304 (0.04%)	0.385	0.128	0.007		
7	lib_mbedtls	86979667 (0.005%)	2482673 (0.028%)	0.285	0.095	0.018		
8	lib_mbedtls	72440941 (0.004%)	976185 (0.011%)	0.135	0.045	0.009	0.007	0.017
9	lib_mbedtls	53966553 (0.003%)	2015969 (0.022%)	0.374	0.125	0.022	0.034	0.176
10	lib_mbedtls	45639841 (0.003%)	8068894 (0.09%)	1.768	0.589	0.003		
11	lib_mbedtls	40299059 (0.002%)	0 (0%)	0				
12	lib_mbedtls	37209040 (0.002%)	9069077 (0.101%)	2.437	0.812	0.004		
13	lib_mbedtls	37061872 (0.002%)	8203147 (0.091%)	2.213	0.738	0.025		
14	lib_mbedtls	36663185 (0.002%)	1684220 (0.019%)	0.462	0.154	0.001		
15	lib_mbedtls	34318559 (0.002%)	2084425 (0.023%)	0.610	0.203	0.021	0.018	0.132
16	lib_mbedtls	33272050 (0.002%)	37539533 (0.419%)	11.283	3.761	0.001		
17	lib_mbedtls	29402620 (0.002%)	17852907 (0.199%)	6.150	2.090	0.001		
18	lib_mbedtls	27391762 (0.002%)	511096 (0.006%)	0.187	0.062	0.000	0.002	0.048
19	lib_mbedtls	25795688 (0.001%)	0 (0%)	0				
20	lib_mbedtls	25215254 (0.001%)	28454223 (0.317%)	11.285	3.762	0.009		
21	lib_mbedtls	23648434 (0.001%)	16272417 (0.181%)	6.981	2.294	0.002		
22	lib_mbedtls	22796214 (0.001%)	2569505 (0.029%)	1.127	0.376	0.014	0.008	0.032
23	lib_mbedtls	22034855 (0.001%)	0 (0%)	0				

图 5.4.4

Configuration Collection Log Summary Function Timeline								
Grouping: Module/Function/Callstack		Flame Graph						
	Module/Function/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTBBuBTB_Rate	PC_Write
1	total	1745109198135	8985245010	0.051	0.017	0.005	0.003	0.007
2	lib_mbedtls	1742915649938 (99.874%)	8702552279 (96.862%)	0.050	0.017	0.004	0.003	0.007
3	lib_mbedtls	2142975625 (0.123%)	243931362 (2.715%)	1.138	0.379	0.012	0.018	0.074
4	lib_mbedtls	12755169 (0.001%)	202186 (0.002%)	0.159	0.053	0.009	0.018	0.019
5	lib_mbedtls	1929590 (0%)	313648 (0.003%)	1.625	0.542			
6	lib_mbedtls	1529527 (0%)	0 (0%)	0				
7	lib_mbedtls	89836 (0%)	0 (0%)	0		0.047		
8	lib_mbedtls	0 (0%)	0 (0%)					
9	lib_mbedtls	0 (0%)	0 (0%)					
10	lib_mbedtls	0 (0%)	0 (0%)					
11	lib_mbedtls	0 (0%)	0 (0%)					
12	lib_mbedtls	0 (0%)	0 (0%)					
13	lib_mbedtls	0 (0%)	0 (0%)					
14	lib_mbedtls	0 (0%)	0 (0%)					
15	lib_mbedtls	0 (0%)	0 (0%)					

图 5.4.5

Configuration Collection Log Summary Function Timeline								
Grouping: Thread/Function/Callstack Flame Graph								
	Thread/Function/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTuBTB_Rate	PC_Write
1	total	1745109198135	8985245010	0.051	0.017	0.005	0.003	0.007
2	matix gcc(TID: 57518)	149310038805 (8.556%)	507149800 (5.644%)	0.034	0.011	0.005	0.003	0.005
3	matix gcc(TID: 57820)	135279134241 (7.752%)	596232538 (6.639%)	0.041	0.014	0.004	0.003	0.006
4	matix gcc(TID: 57813)	133835602590 (7.669%)	520707452 (5.795%)	0.039	0.013	0.005	0.003	0.005
5	matix gcc(TID: 57823)	132417067076 (7.580%)	542531790 (6.038%)	0.041	0.014	0.005	0.003	0.006
6	matix gcc(TID: 57816)	126558055696 (7.252%)	547027263 (6.089%)	0.043	0.014	0.005	0.003	0.006
7	matix gcc(TID: 57814)	126548812522 (7.252%)	543919400 (6.053%)	0.043	0.014	0.004	0.003	0.005
8	matix gcc(TID: 57815)	118956422558 (6.817%)	526253790 (5.857%)	0.044	0.015	0.005	0.003	0.006
9	matix gcc(TID: 57809)	110990144782 (6.36%)	554478737 (6.171%)	0.050	0.017	0.004	0.003	0.006
10	matix gcc(TID: 57810)	102372034519 (5.869%)	577818152 (6.429%)	0.056	0.019	0.004	0.002	0.007
11	matix gcc(TID: 57819)	96272056806 (5.517%)	552041205 (6.144%)	0.057	0.019	0.005	0.003	0.008
12	matix gcc(TID: 57824)	94090411034 (5.392%)	567123209 (6.315%)	0.060	0.020	0.005	0.003	0.008
13	matix gcc(TID: 57821)	88476252536 (5.07%)	561744091 (6.252%)	0.063	0.021	0.006	0.003	0.009
14	matix gcc(TID: 57812)	87529292338 (5.016%)	580885999 (6.455%)	0.066	0.022	0.005	0.003	0.009
15	matix gcc(TID: 57822)	86137424991 (4.936%)	541517588 (6.027%)	0.063	0.021	0.005	0.003	0.009
16	matix gcc(TID: 57817)	77930210343 (4.466%)	547794263 (6.087%)	0.070	0.023	0.005	0.003	0.010
17	matix gcc(TID: 57811)	77916373847 (4.465%)	549746655 (6.119%)	0.071	0.024	0.005	0.003	0.009
18	matix gcc(TID: 57808)	492025439 (0.282%)	208180890 (2.317%)	4.231	1.410	0.006		0

图 5.4.6

Configuration Collection Log Summary Function Timeline Zoomview								
Grouping: Core/Function/Callstack Flame Graph								
	Core/Function/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTuBTB_Rate	PC_Write
1	total	940280519650	6318995827	0.067	0.022	0.003	0.002	0.008
2	lib 1	70495957102 (7.427%)	578994420 (9.131%)	0.082	0.027	0.002	0.001	0.009
3	lib 48	59134675166 (6.229%)	238510680 (3.775%)	0.040	0.013	0.003	0.002	0.005
4	lib 14	58915285757 (6.266%)	491340026 (7.776%)	0.083	0.028	0.002	0.001	0.009
5	lib 32	57183355132 (6.024%)	246290348 (3.896%)	0.043	0.014	0.003	0.002	0.006
6	lib 25	50055249454 (5.273%)	342766683 (5.424%)	0.068	0.023	0.002	0.001	0.008
7	lib 23	48718758370 (5.132%)	342271878 (5.417%)	0.070	0.023	0.002	0.001	0.008
8	lib 33	46819463967 (4.932%)	194899607 (3.084%)	0.042	0.014	0.003	0.002	0.005
9	lib 15	45411553812 (4.794%)	369700340 (5.851%)	0.081	0.027	0.002	0.001	0.009
10	lib 34	39806773785 (4.193%)	188150245 (2.961%)	0.042	0.014	0.004	0.002	0.005
11	lib 35	36503253180 (4.056%)	155239779 (2.457%)	0.040	0.013	0.003	0.002	0.005
12	lib 16	36389573729 (3.836%)	256209902 (4.055%)	0.070	0.023	0.002	0.001	0.008
13	lib 29	36115633013 (3.805%)	264613712 (4.188%)	0.073	0.024	0.002	0.001	0.009
14	lib 17	35958359788 (3.789%)	244580648 (3.871%)	0.068	0.023	0.002	0.001	0.008
15	lib 27	31388602044 (3.306%)	234570453 (3.712%)	0.075	0.025	0.002	0.001	0.009
16	lib 30	29715162701 (3.133%)	219579356 (3.475%)	0.074	0.025	0.002	0.001	0.009
17	lib 49	28601342423 (3.013%)	119528226 (1.892%)	0.042	0.014	0.004	0.002	0.005
18	lib 9	27742207639 (2.922%)	242436965 (3.837%)	0.087	0.029	0.002	0.001	0.010
19	lib 28	26449336489 (2.789%)	195959748 (3.101%)	0.074	0.025	0.003	0.002	0.009
20	lib 31	23274634928 (2.452%)	174721165 (2.769%)	0.075	0.025	0.003	0.001	0.009
21	lib 24	22602785238 (2.413%)	165332397 (2.616%)	0.072	0.024	0.003	0.002	0.008
22	lib 18	19126488527 (2.015%)	138946887 (2.199%)	0.073	0.024	0.003	0.002	0.009
23	lib 43	13383491321 (1.411%)	64185310 (1.015%)	0.048	0.016	0.006	0.002	0.007

图 5.4.7

提供 4 种不同组合方式来查看，Core/Thread/Module/Function/Callstack 与 top-down 指标的关系（如图 5.4.8）。

- Grouping Function/Callstack
- Grouping Module/Function/Callstack
- Grouping Thread/Function/Callstack
- Grouping Core/Function/Callstack
- 对于采集 java 程序，还支持按照 Grouping class/Method/Callstack

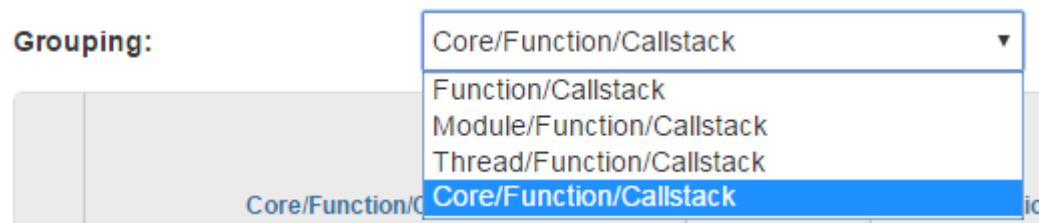


图 5.4.8

辅助功能

- Grouping 结果可以上下折叠

- 基于选中 PMU 指标排序
- PMU 指标可左右折叠
- 隐藏选中 PMU 指标
- 导出当前界面结果到 EXCEL 文件
- 支持 Function 分页显示，当前页面数据行数可选

5.4.3 Timechart 分析

➤ 查看 PMU 指标时序图（如图 5.4.9）

基于选中的 PMU 指标（最多 4 个），显示其时序图。可分析不同指标在整个数据采集过程中的波形走势，同一时刻不同指标的关联影响，以及选定一段时间内的 top-down 整体值、TopN 热点函数列表、波形等数据变化。

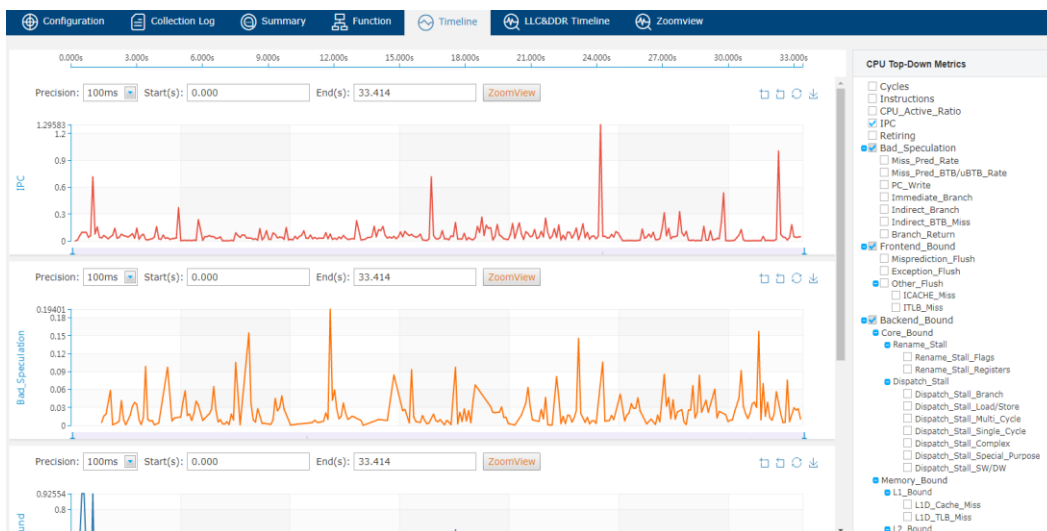


图 5.4.9

➤ 辅助功能

- 基于任一 PMU 指标时序图窗口放大（如图 5.4.10），选中标记处鼠标左键在时序图中选中指定时间区域，自动放大此时间区域的时序图。

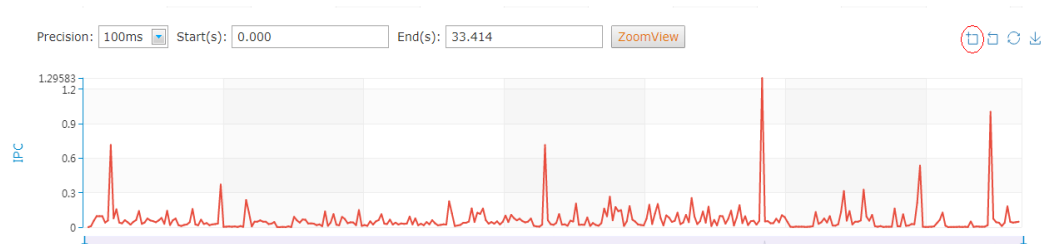


图 5.4.10

- 基于任一 PMU 指标时序图选择窗口滑动分析（如图 5.4.11），选中标记处鼠标左右拖动。



图 5.4.11

- 波形图可以支持不同的插值步长，默认为 100ms（如图 5.4.12）。



图 5.4.12

5.4.4 Zoomview 分析

- 查看圈定时间内的热点函数及指标时序图（如图 5.4.13）

在 Timechart 页签点击如下红圈中的 zoom 按钮，然后在波形图上鼠标左键框选时间段范围，再点击 ZoomView 按钮，会弹出 Zoomview 页签，并生成对应时间段的分析。

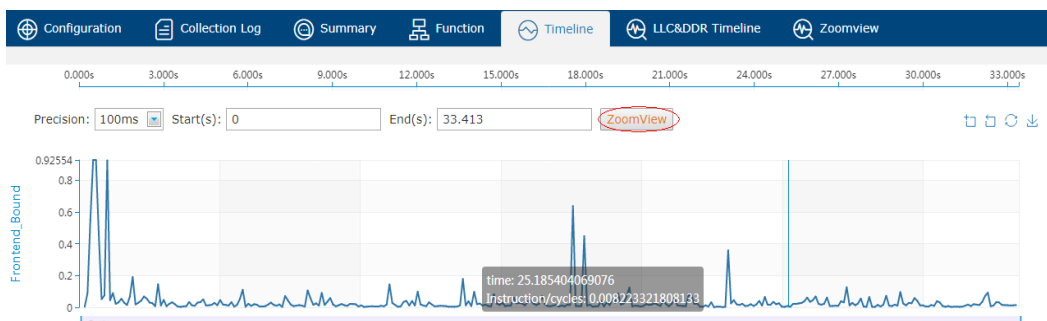


图 5.4.13

- Zoomview 页签

Zoomview 页签包括左侧的 Zoom List 区域，上部的 Function 区域和下部的 Timechart 区域（如图 5.4.14），其中 Zoom List 区域为选择的时间段对应的分析名称，如名称“time-0-33413”表示从 0 秒到 33.413 秒这段时间内的分析，Function 区域为选择时间段内的热点函数，Timechart 区域为选择时间段的波形图。

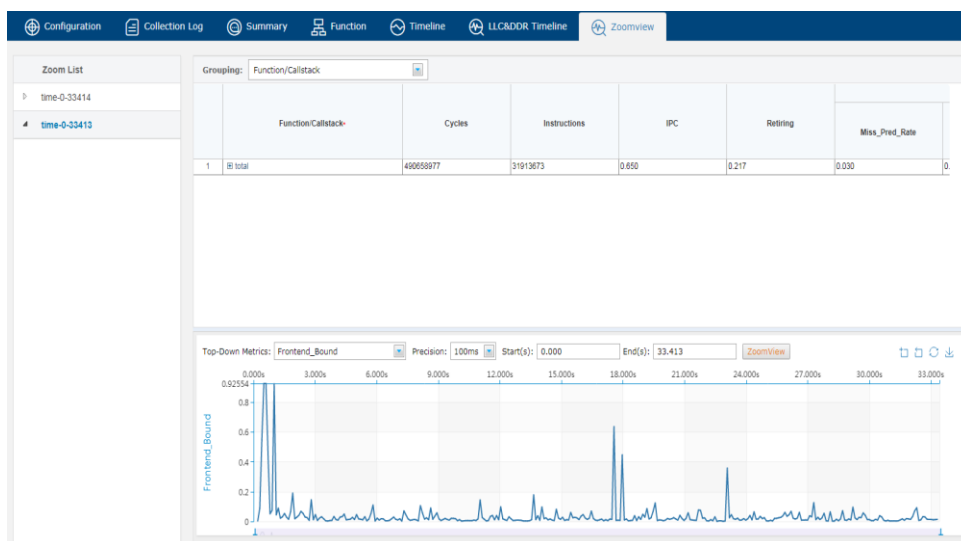


图 5.4.14

可以在 Zoomview 页签生成的波形图上，继续选择时间段生成次一级分析，方法跟 Timechart 页签生成时间段分析相同，此时的次一级分析名称显示在一级分析名称下面（如图 5.4.15）。

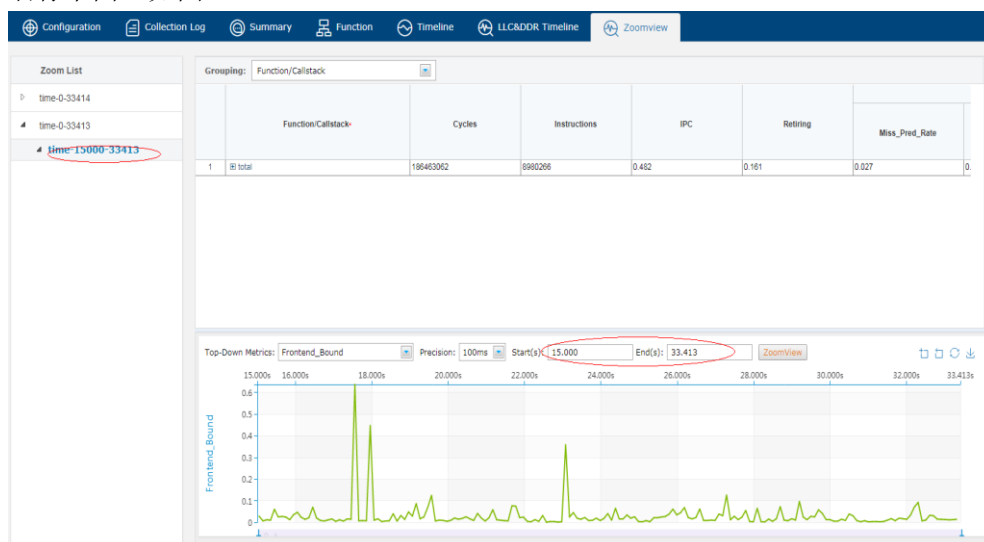


图 5.4.15

5.4.5 Code 分析

- 查看热点函数的代码映射（如图 5.4.16），通过该功能可以分析热点函数内部的热点指令，热点指令即指函数内 Cycles 事件占比最高的 Top 指令，功能还支持查看热点指令对应的高级语言文件及行号。功能还支持对汇编代码进行控制流分析，通过划分 basic block 并标示出跳转关系及颜色，可以清晰看到各个汇编代码块的“热度”。

在 Function 页签双击函数名称，会跳转到 Code 页签，Code 页签分为几个区域：

- 左侧为 Function List，是模块名及模块内的函数的列表
- 右侧上部包括：HardWare Event 下拉菜单，支持显示不同的 HardWare 事件，Total Count 表示该事件的求和值，File Name 表示函数所在的源文件

- 右侧下部包括函数源代码、函数汇编代码和函数汇编代码 **basic block** 的控制流图。在源代码和汇编代码区域，计算出每行源代码和汇编代码对应的 **HardWare** 事件计数值及占该事件 **Total Count** 的百分比（值为空的表示 0），百分比最高的即为函数的热点指令，在控制流图区域，通过图形标示出 **basic block** 的跳转关系和颜色，可以直观看到函数内的热点及调用关系。

在 Code 页签的源代码区域，包括：

- **Source Line**: 源代码行号
- **Source**: 改行对应的源代码
- **Count(Percent)**: 该行源代码对应的 **PMU** 事件计数值及占该事件 **Total Count** 的百分比
- 过滤框: 过滤出包含指定字符串的源码

在 Code 页签的汇编代码区域，包括：

- **Address**: 汇编指令地址
- **Source Line**: 汇编指令对应的高级代码行号
- **Assembly**: 汇编指令
- **Count(Percent)**: 该行汇编代码对应的 **PMU** 事件计数值及占该事件 **Total Count** 的百分比
- 过滤框: 过滤出包含指定字符串的汇编代码
- 折叠: 通过点击统一打开或者折叠所有 **basic block**
- 显示/取消显示汇编码的 **File**: 通过点击统一显示或者取消显示汇编对应 **File**
- 支持汇编码下载

辅助功能

在源代码、汇编代码或控制流图上任意区域单击，三者可以联动高亮：

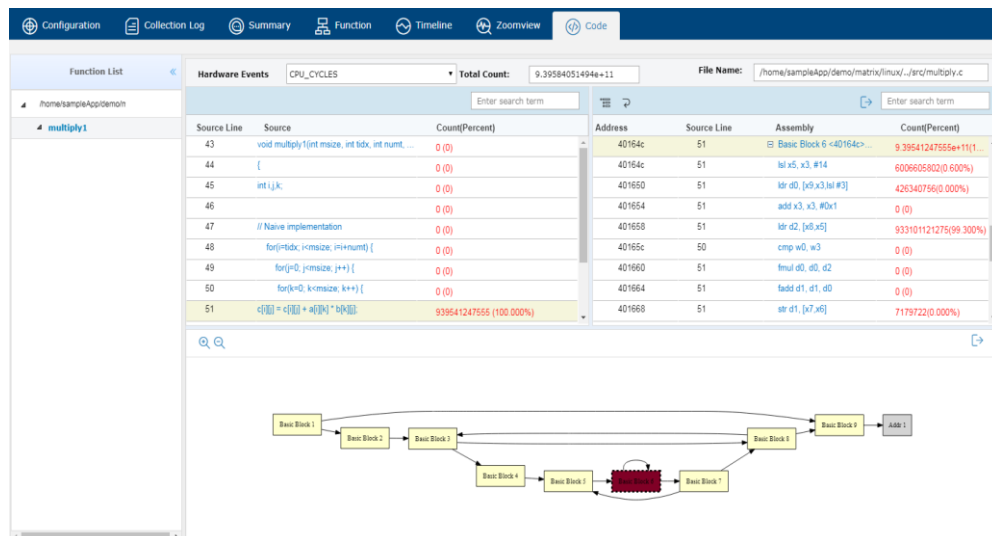


图 5.4.16

对于采样类型为 **Java Mixed-Mode** 采集到的 **java** 的方法，因为 **java** 的 **jit** 编译的限制，当前只有对应的源码，没有汇编和控制流图区域，如图 5.4.16.1：

Function List

«

/tmp/perf-13422.map

4 LRunner\$Work::doV

Hardware Events

CPU_CYCLES

Total Count:

4.9342260644e+10

File Name:

/home/sampleApp/examples/Runner.java

Enter search term

Source Line	Source	Count(Percent)
21	public int doWork(int iterations, Executor executor, int countDown, CountDow...	0(0)
22	int sum = 0;	0(0)
23	for (int i = 0; i < iterations; i++) {	59197739(0.12%)
24	int index = rand.nextInt(localState.length);	47513527403(96.294%)
25	sum += localState[index];	943297026(1.912%)
26	localState[index] = i;	0(0)
27	}	0(0)
28	if (countDown > 0) {	72036732(0.146%)
29	// submit next message	0(0)
30	executor.execute(() -> doWork(iterations, executor, countDown-1, latch));	669967857(1.358%)
31	} else {	0(0)
32	// finished	0(0)
33	latch.countDown();	0(0)
34	}	0(0)
35	return sum;	0(0)
36	}	0(0)

图 5.4.16.1

对于采集 java 程序，为了使能映射的源码更精确，建议开启 -XX:+UnlockDiagnosticVMOptions -XX:+DebugNonSafepoints 选项。

5.4.6 Resource 分析

- 查看采样时间后台 CPU 各个核的调度情况（如图 5.4.17）
显示采样时间内各个 CPU 核的运行调度情况，时间精度最小是微妙。将鼠标悬停在图上任意一点，显示当前时刻 CPU 核的运行状态，包括 idle 和 running 这 2 种状态，并显示状态起止时间和持续时间，对于 running 状态显示当前运行的进程名字和进程 id。其中 idle 状态表示 CPU 空闲，running 状态表示 CPU 正在运行程序。
 - Filter 按钮：点击弹出过滤对话框，可以过滤掉不关注的 CPU 核，如图 5.4.18
 - Function 按钮：选中 Function 按钮后，页签下面出来 Function 表格，显示选中的 CPU 在选中时间段内的热点函数
 - reset 按钮：如图 5.4.17 红圈，时序图放大缩小后，点该按钮恢复原始精度
 - 左/右箭头：沿着选中时间点及选中 CPU 核，向上或下一个状态单步前进
 - 上/下箭头：固定选中时间点，在上或下一个 CPU 核上切换
 - 放大/缩小按钮：放大/缩小时序图的，当前最小精度为微妙级

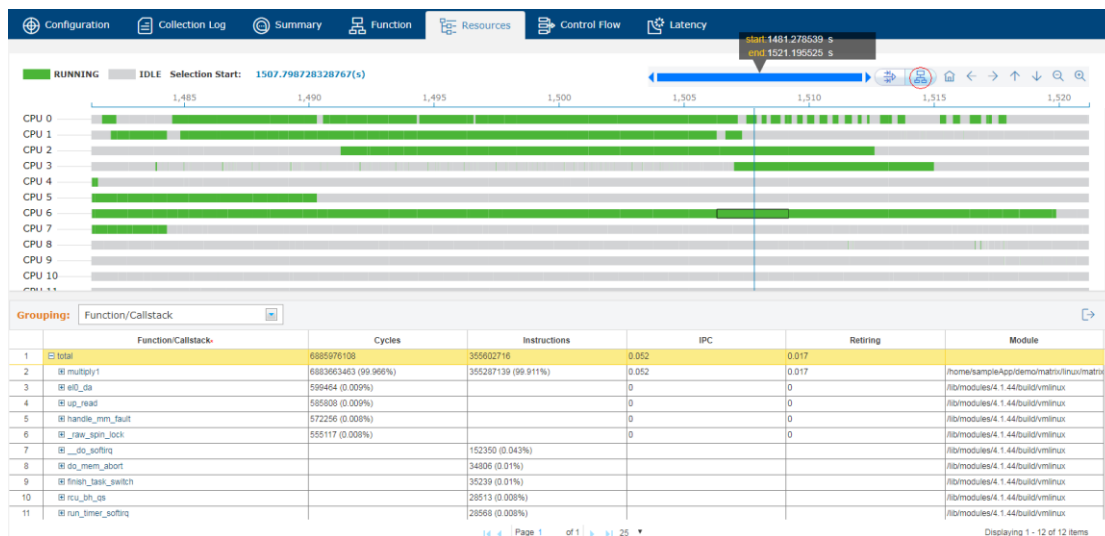


图 5.4.17

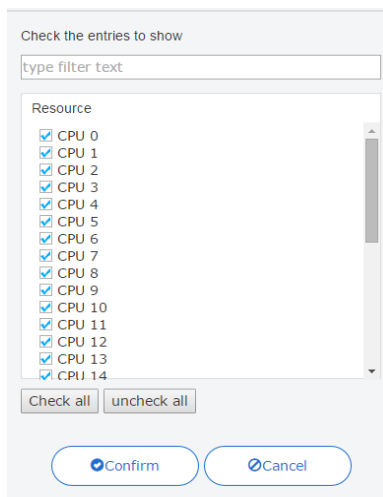


图 5.4.18

5.4.7 Control Flow 分析

➤ 查看采样时间后台进程的调度情况（如图 5.4.19）

显示采样时间内各个进程的运行调度情况，时间精度最小是微秒。将鼠标悬停在图上任意一点，显示当前时刻该进程的运行状态，包括 **wait_blocked**、**wait_for_cpu** 和 **running** 这 3 种状态，并显示状态起止时间和持续时间。其中 **wait_blocked** 状态表示进程阻塞无法运行，**wait_for_cpu** 状态表示进程处于可运行队列中，**running** 状态表示进程正在运行。

- **Filter** 按钮：点击弹出过滤对话框，可以过滤掉不关注的进程，如图 5.4.20
- **Function** 按钮：选中 **Function** 按钮后，页签下面出来 **Function** 表格，显示选中的线程在选中时间段内的热点函数
- **reset** 按钮：如图 5.4.19 红圈，时序图放大缩小后，点该按钮恢复原始精度
- 左/右箭头：沿着选中时间点及选中进程，向上或下一个状态单步前进
- 上/下箭头：固定选中时间点，在上或下一个进程切换
- 放大/缩小按钮：放大/缩小时序图的，当前最小精度为微妙级

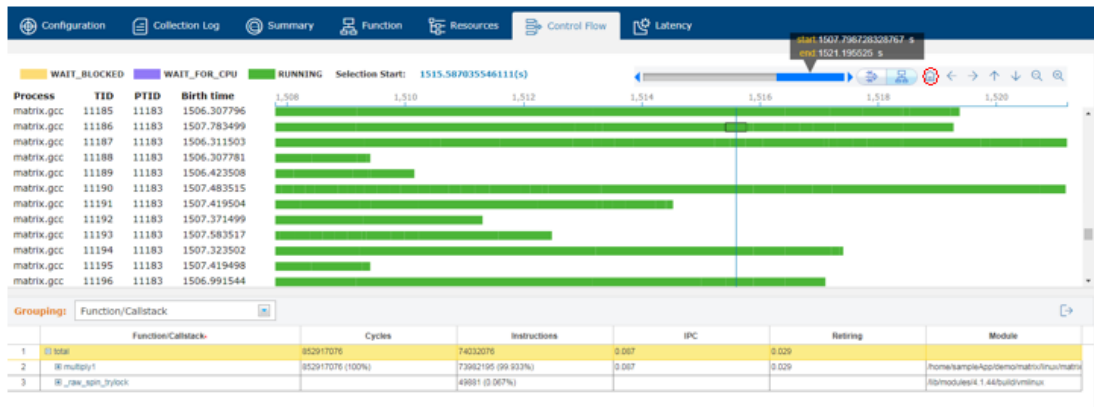


图 5.4.19

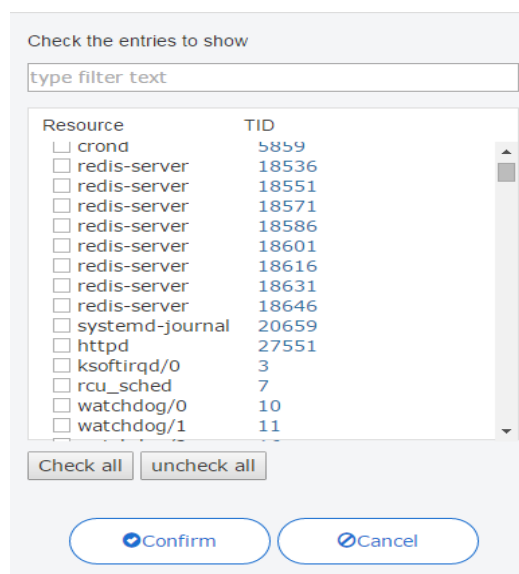


图 5.4.20

5.4.8 Latency 分析

- 查看采样时间后台进程的调度延迟情况（如图 5.4.21）

调度延迟即进程进入运行队列到获取处理器执行之间的时间差。可以统计采集期间所有进程的切换次数、平均调度延迟、最大调度延迟和最大延迟的时间点。

支持按照 Task 名字进行过滤和导出 excel 格式的数据。

Configuration Collection Log Summary Function Resources Control Flow Latency					
Task: Enter search term					
NO.	Task	Switches	Average delay (ms)	Maximum delay (ms)	Maximum delay at (s)
1	kworker/39:1:1872	5	3.962400	4.002000	1482.335487
2	kworker/56:1:1887	3	3.937000	4.001000	1483.135487
3	kworker/6:1:1934	19	3.780737	4.002000	1490.731486
4	kworker/58:0:10635	22	3.608045	4.000000	1509.107488
5	kworker/5:1:1914	8	3.492750	4.002000	1489.743485
6	kworker/20:2:4311	19	3.365368	4.004000	1485.563488
7	kworker/40:1:1873	23	3.285174	4.002000	1501.323488
8	kworker/21:1:1854	20	3.191800	4.004000	1486.551488
9	kworker/49:1:1880	13	3.074385	4.001000	1513.215488
10	kworker/37:2:4244	15	2.914800	4.000000	1486.119485
11	kworker/22:1:1856	11	2.904909	4.001000	1490.539486
12	kworker/50:1:1882	18	2.873833	4.002000	1497.203487
13	kworker/53:1:1883	17	2.820588	4.000000	1500.167487
14	kworker/42:2:4309	19	2.720368	4.002000	1482.299487
15	kworker/51:1:1881	6	2.651167	3.998000	1492.191485
16	kworker/17:1:1851	23	2.607348	4.001000	1503.599485
17	kworker/1:1:1845	19	2.525842	4.001000	1489.791485
18	kworker/32:2:4245	40	2.400850	4.000000	1497.419487
19	kworker/0:0:4	10	2.399400	4.002000	1497.807489
20	kworker/33:1:1866	37	2.205108	4.001000	1496.407488

图 5.4.21

5.4.9 Disk IO 分析

➤ 查看采样时间 disk io 的情况（如图 5.4.22）

Thread: 查看采样时间每个线程的 I/O Wait 和 I/O APIs 时序图

I/O Queue Depth: 查看采样时间每个磁盘的 I/O 请求队列长度变化时序图

I/O Operation: 查看采样时间每个磁盘的 I/O 操作时序图

I/O Transfer: 查看采样时间每个磁盘的 I/O 操作数据大小时序图

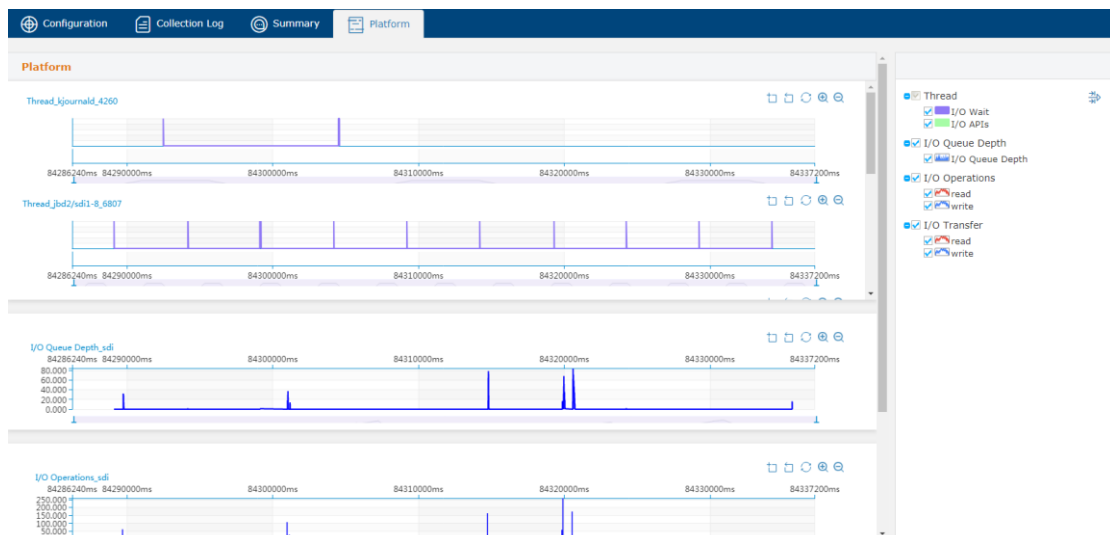


图 5.4.22

5.4.10 Locks and Waits 分析

➤ 查看采样时间程序锁和等待的情况（如图 5.4.23）

界面分为 2 部分，上部为表格，下部为时序图。上部的表格对采集到的数据从 Function/Callstack 、 SyncObject/Function/Callstack 、 Thread/Function/Callstack 和

Module/Function/Callstack 这 4 个维度进行分组展示, 表格的 Wait Time 列表示等待时长, Wait Count 列表示等待次数。通过双击函数名, 可以跳转到 Locks and Waits Code 页面。下部为进程/线程时序图, 其中黄色表示线程因为调用 glibc 的 sleep、usleep、mutex、cond、spinlock、rwlock 或 semaphore 接口导致进入了阻塞状态, 通过鼠标悬浮于时序图上, 可以看到引起程序阻塞的函数及其调用栈。

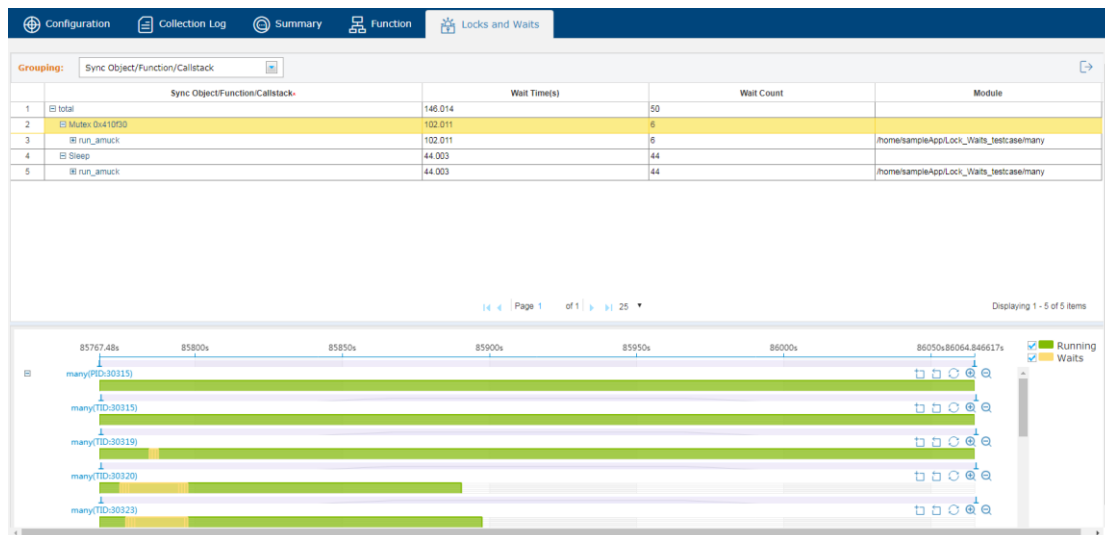


图 5.4.23

5.4.11 Locks and Waits Code 分析

- 查看采样时间程序锁和等待的情况 (如图 5.4.24)

通过在 Locks and Waits 页面双击函数, 可以跳转到该页面。该页面可以展示程序中调用了 glibc 的 sleep、usleep、mutex、cond、spinlock、rwlock 或 semaphore 接口的函数及其反汇编。如下, 左侧源代码区域展示了该函数调用 sleep(1)阻塞的时间和次数, 右侧汇编代码区域展示了跳转指令到 sleep 函数的阻塞时间和次数。

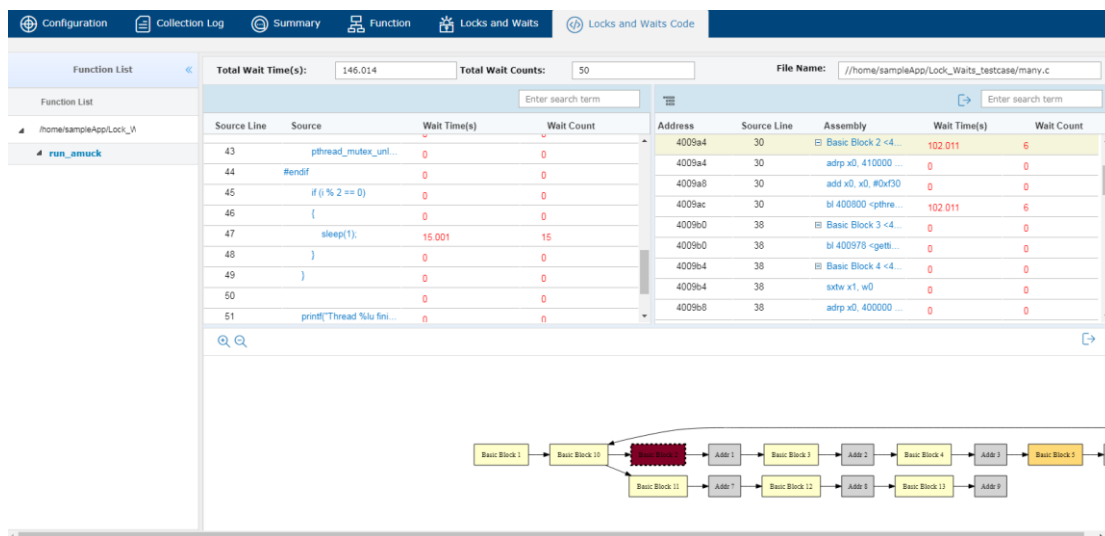


图 5.4.24

5.4.12 LLC&DDR 分析

1) Hisilicon CPU Hi1616 环境

注: 为了使用 LLC&DDR 分析功能, 需要首先编译、安装对应内核版本的内核驱动程序(具体请依据源码中 source/llc-ddr/Hi1616_4P_DFX 目录下的 README 说明)

➤ 查看采样时间 LLC&DDR 的汇总情况 (如图 5.4.25)

下图展示了汇总的分析数据, 指标包括 CPU 每个 die 的 llc 读写带宽及命中率, 及 CPU 的 ddr 内存采集时间的读写带宽。

LLC and DDR statistics						
Metric		Die0	Die1	Die2	Die3	Total
llc_rd	Hit Bandwidth (MB/s)	3.316	1.31	2.731	1.537	8.994
	Bandwidth (MB/s)	6.541	2.606	5.276	3.059	17.482
	Hit_Rate	50.703	50.259	51.749	50.229	50.245
llc_wr	Hit Bandwidth (MB/s)	0.129	0.109	0.006	0.007	0.251
	Bandwidth (MB/s)	0.254	0.215	0.012	0.011	0.492
	Hit_Rate	50.906	50.904	55.501	64.262	63.636
ddr_rd	Bandwidth (MB/s)	0.505	0.446	0.453	0.441	1.845
	Bandwidth (MB/s)	0.426	0.165	0.079	0.089	0.759

图 5.4.25

➤ 查看采样时间 LLC&DDR 的时间分布情况 (如图 5.4.26)

下图展示了 CPU 每个 die 的 llc&ddr 的读、写命中率及带宽, 左侧主轴表示带宽, 单位是 MB/s, 右侧副轴表示命中率, 单位是百分比。

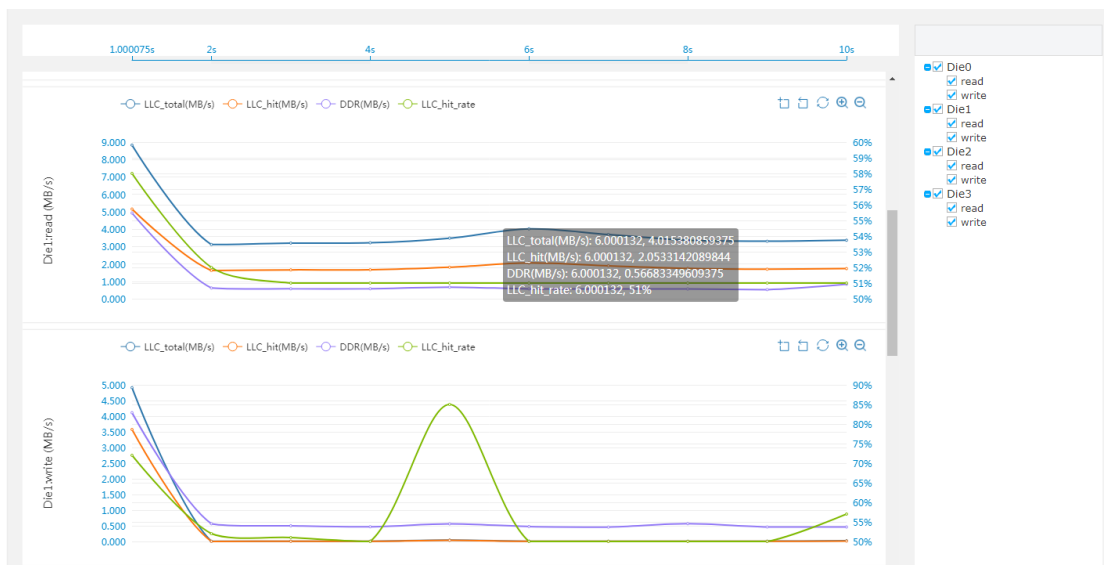


图 5.4.26

2) Hisilicon CPU Hisilicon 1620 环境

注: 为了使用 LLC&DDR 分析功能, 需要首先在 1620 环境检查当前内核的 CONFIG_HISI_PMU 开关是否打开(确认开关打开方式请使用 `zcat /proc/config.gz|grep HISI_PMU`)

针对 Summary (如图 5.4.27) 和 Timeline (如图 5.4.28) 页签额外支持 DieX_Out_Hit(MB/s)、DieX_Out_Total(MB/s)、DieX_Out_Hit_Rate(MB/s)和 HHA; 具体包括三部分如下:

➤ 查看采样时间 LLC&DDR&HHA 的汇总情况 (如图 5.4.27)

下图展示每个 Die 上 DDR 和 L3C 的在采样周期内总的读写带宽，以及每个 Die 上的 L3C 的命中率和访问数据的命中带宽，带宽均已 MB/s 为单位；另包含关于 HHA 的 8 个指标，关于本地 HHA 接收的操作数量以及 HHA 接收到来自片外的操作数量等数据。（以上数据均为采样周期内的数据总和）。

LLC and DDR statistics						
Metric		Die1	Die3	Die5	Die7	Total
ddr_rd	Bandwidth (MB/s)	9.008	197.605	6.271	10.494	223.378
ddr_wr	Bandwidth (MB/s)	3.728	35.113	3.642	3.649	46.132
l3c_rd	Hit Bandwidth (MB/s)	1.964	51.215	1.279	2.571	57.029
	Bandwidth (MB/s)	14.494	210.614	11.124	14.88	251.113
	Hit_Rate	0.136	0.243	0.115	0.173	0.227
	Out_Hit Bandwidth (MB/s)	5.046	27.058	4.627	5.487	42.219
	Out_Bandwidth (MB/s)	14.718	219.612	11.148	15.153	260.63
	Out_Hit_Rate	0.343	0.123	0.415	0.362	0.162
l3c_wr	Hit Bandwidth (MB/s)	0	0	0	0	0
	Bandwidth (MB/s)	0	0	0	0	0
	Hit_Rate	0	0	0	0	0
	Out_Hit Bandwidth (MB/s)	0	0	0	0	0
	Out_Bandwidth (MB/s)	0	0	0	0	0
	Out_Hit_Rate	0	0	0	0	0
hha	hha_opt_num (counts/s)	192856	5353784	156136	212385	5915161
	hha_out_opt_num (counts/s)	811	2405192	77060	116073	2599136
	hha_die_num (counts/s)	116379	145125	3217	1350	266071
	hha_mediate_num (counts/s)	71951	1054736	34729	44158	1205572
	hha_snoop_num (counts/s)	96275	801675	113141	119424	1130515
	hha_out_snoop_num (counts/s)	2216	233560	67343	67245	370364
	hha_s_dir_counter (counts/s)	0	0	0	0	0
	hha_e_dir_counter (counts/s)	0	0	0	0	0

图 5.4.27

➤ 查看采样时间 LLC&DD&DDR 的时间分布情况（如图 5.4.28）

下图上半部分展示了 CPU 每个 die 的 llc&ddr 的读、写命中率及带宽，左侧主轴表示带宽，单位是 MB/s，右侧副轴表示命中率，单位是百分比；下半部分展示了关于 HHA 的 8 个指标，在采样周期内的每一时刻的数据数值。



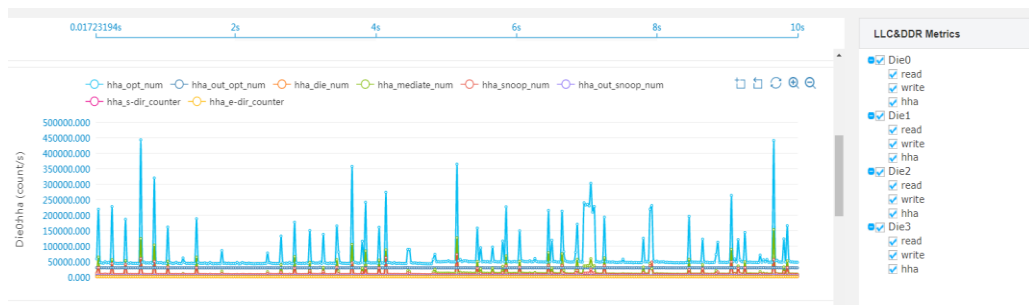


图 5.4.28

5.4.13 Flame Graph 分析

- 查看采样时间热点函数的火焰图（如图 5.4.29），帮助用户直观且迅速找到热点代码路径：

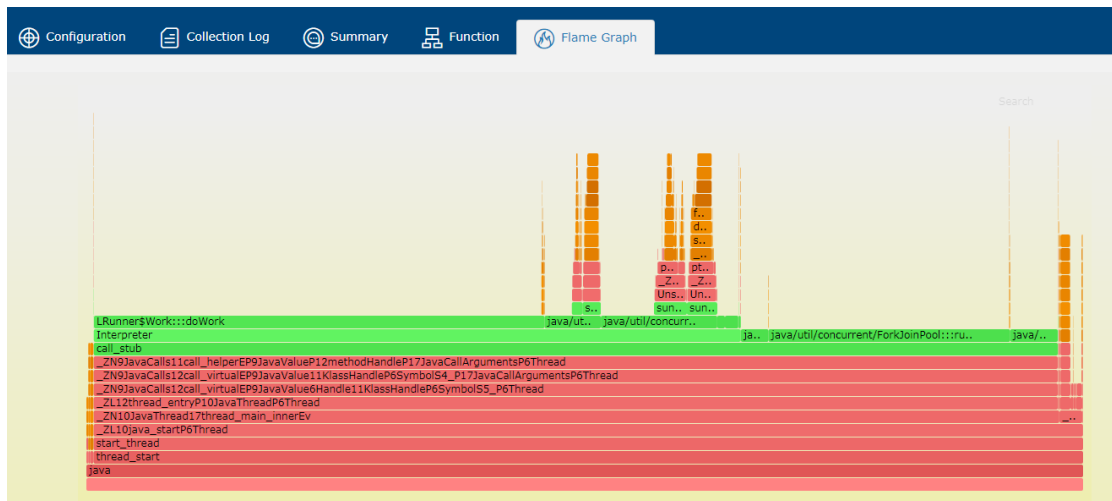


图 5.4.29

5.4.14 Network Input and Output 分析

- 查看采样时间 Network Input and Output 的汇总情况（如图 5.4.30）
下图展示了汇总的分析数据，指标包括每网口收发包带宽、收发包个数、收发包错包个数和收发包丢包个数。

Interfaces				
item	docker0	eth2	eth3	lo
receive(MB/s)	0	0.01	25.223	0
receive_packets	0	117	399226	0
receive_drop_packets	0	0	0	0
receive_error_packets	0	0	20	0
send(MB/s)	0	0	1125.129	0
send_packets	0	0	93396	0
send_drop_packets	0	0	0	0
send_error_packets	0	0	0	0

图 5.4.30

- 查看采样时间的 **Network Input and Output** 时间分布情况（如图 5.4.31）
- 下图展示了每个网口收发包带宽、收发包个数、收发包错包个数和收发包丢包个数随时间的变化情况，左侧主轴表示带宽，单位是 **MB/s**，右侧副轴表报文个数，单位是 **packet**。

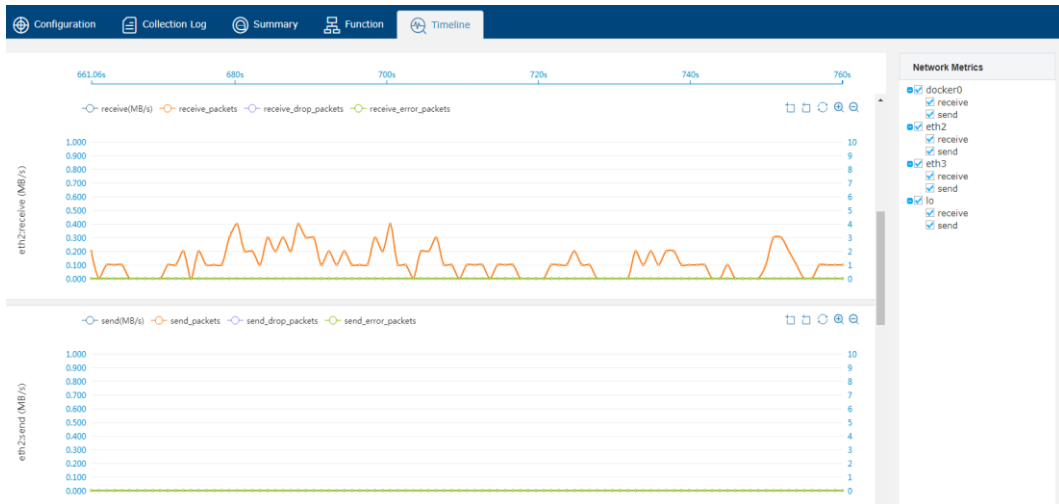


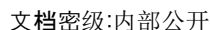
图 5.4.31

5.4.15 False Sharing 分析

- 多进程共享采集分析，下图展示了分析的所有指标数据（如图 5.4.32）。

Hi1620 PMU Events	
Elapsed Time(s)	3.100
Cycles	8898154328
Instructions	2110506420
CPU_Active_Ratio	0.014
IPC	0.712
L2D_CACHE_LD	71240358
L2D_CACHE_ST	61655684
L2D_CACHE_REFILL_LD	46052049
L2D_CACHE_REFILL_ST	36473271
L2D_CACHE_WB_VICTIM	7756956
L2D_CACHE_INVALID	68575888
UNALIGNED_LD_SPEC	82451
UNALIGNED_ST_SPEC	839866
UNALIGNED_LDST_SPEC	925430
LD_SPEC	1034460417
ST_SPEC	362075071
ReadShare	46623652
ReadUnique	37423670
csnp_vld	99140412
csnp_share	30291643
csnp_unique	40042405
memstall_anyload	2975348972
memstall_anystore	20080061
memstall_L1miss	1667202834
memstall_L2miss	1617677581
csnp_unique_rate	1.070
csnp_share_rate	0.809
l2_miss_rate	0.182

图 5.4.32



```
arm44:/home # python /opt/Malluma/Malluma-analyzer/bin64/Malluma-analyzer.py --collect general-exploration sleep 5

Top-down total results:
-----
Elapsed Time(s)                5.09
Cycles                        959467
Instructions                   81076
CPU_Active_Ratio               0.0
IPC                           0.845
Retiring                      0.282
Bad_Speculation                0.019
  Miss_Pred_Rate               0.048
  Miss_Pred_BTBT_uBTBT_Rate    None
  PC_Write                     0
  Immediate_Branch             0
  Indirect_Branch              0
  Indirect_BTBT_Miss           None
  Branch_Return                0
Frontend_Bound                 0.904
  Misprediction_Flush          0.512
  Exception_Flush              0.002
  Other_Flush                  0.39
  ICACHE_Miss                  None
  ITLB_Miss                    None
Backend_Bound                  -0.204
  Core_Bound                   0
    Rename_Stall               0
      Rename_Stall_Flags       0
      Rename_Stall_Registers   0
    Dispatch_Stall             0
      Dispatch_Stall_Branch     0
      Dispatch_Stall_Load_Store 0
      Dispatch_Stall_Multi_Cycle 0
      Dispatch_Stall_Single_Cycle 0
      Dispatch_Stall_Complex    0
      Dispatch_Stall_Special_Purpose 0
      Dispatch_Stall_SW_DW      0
  Memory_Bound                 0
    L1_Bound
      L1D_Cache_Miss           None
      L1D_TLB_Miss             None
    L2_Bound
      L2_Cache_Miss            None
      L2_TLB_Miss              None
    L1D_Bandwidth(MB/s)         0
    L2_Bandwidth(MB/s)          0
    Bus_Bandwidth_RD(MB/s)      0
    Bus_Bandwidth_WR(MB/s)      0
-----
arm44:/home #
```

```
server-for-toolsdevelo:/opt/Malluma-v0.4/Malluma-analyzer/bin64 # python malluma-analyzer --collect scheduling -d 5
```

Task	Switches	Average delay ms	Maximum delay ms	Maximum delay at
kworker/28:1H:3659	1	avg: 0.000010 ms	max: 0.000010 ms	max at: 4698748.569617 s
kworker/3:1:17753	1	avg: 0.000009 ms	max: 0.000009 ms	max at: 4698744.876867 s
kworker/17:2:17052	1	avg: 0.000009 ms	max: 0.000009 ms	max at: 4698744.708864 s
kworker/28:1:4461	5	avg: 0.000008 ms	max: 0.000017 ms	max at: 4698745.580868 s
kworker/20:2:27716	1	avg: 0.000008 ms	max: 0.000008 ms	max at: 4698744.672865 s
kworker/23:2:13329	1	avg: 0.000008 ms	max: 0.000008 ms	max at: 4698744.632865 s
perf:3884	2	avg: 0.000007 ms	max: 0.000009 ms	max at: 4698743.700034 s
kworker/5:0:9613	4	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698747.848858 s
kworker/6:2:24831	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.840858 s
kworker/4:0:11829	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.864868 s
kworker/24:0:29220	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.620867 s
kworker/2:1:16589	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.888865 s
kworker/21:2:28590	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.656867 s
kworker/u64:2:1779	6	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698744.524863 s
kworker/16:2:2556	6	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698743.700863 s
kworker/25:0:7889	2	avg: 0.000006 ms	max: 0.000007 ms	max at: 4698744.608863 s
kworker/19:2:14151	1	avg: 0.000006 ms	max: 0.000006 ms	max at: 4698744.684860 s
khugepaged:570	1	avg: 0.000006 ms	max: 0.000006 ms	max at: 4698746.720857 s
kworker/7:1:29364	2	avg: 0.000006 ms	max: 0.000008 ms	max at: 4698745.828864 s
kworker/1:1:5318	2	avg: 0.000005 ms	max: 0.000007 ms	max at: 4698744.900866 s
kworker/8:1:9803	5	avg: 0.000005 ms	max: 0.000006 ms	max at: 4698745.580857 s
kworker/0:1:17402	2	avg: 0.000005 ms	max: 0.000007 ms	max at: 4698744.912864 s
watchdog/15:81	2	avg: 0.000005 ms	max: 0.000005 ms	max at: 4698744.248852 s
watchdog/5:31	2	avg: 0.000005 ms	max: 0.000005 ms	max at: 4698744.208854 s
watchdog/2:16	2	avg: 0.000005 ms	max: 0.000006 ms	max at: 4698744.196855 s
kjournald:3664	4	avg: 0.000004 ms	max: 0.000006 ms	max at: 4698748.576917 s
watchdog/25:131	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.288854 s
irqbalance:5470	1	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.624029 s
migration/6:37	1	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698743.700025 s
kworker/18:1:27051	2	avg: 0.000004 ms	max: 0.000007 ms	max at: 4698745.696860 s
watchdog/16:86	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.252854 s
watchdog/4:26	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.204856 s
watchdog/21:111	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.272854 s
watchdog/24:126	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.284856 s
watchdog/0:10	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.188856 s
watchdog/20:106	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.268856 s
watchdog/3:21	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.200854 s
watchdog/27:141	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.296854 s
watchdog/29:151	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.304853 s
iostat:3883	5	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.550118 s
kworker/22:2:31206	5	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.644856 s
watchdog/1:11	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.192854 s
watchdog/18:96	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.260853 s

图 6.2

6.2 命令参数

➤ 数据采集 Collect 相关:

参数	说明
--collect=ANALYSIS_TYPE	Specify analysis type, example. --collect general-exploration or --collect scheduling. Could specify sampling gpu. Example: --collect genera- exploration --gpu-events
-r DIRECTORY, --result-dir=DIRECTORY	Specify the directory used for creating the data collection results,example ./r000ge or ./r000sch.
-p PID, --target-pid=PID	Specify ID for the process to which data collection should be attached.
--system-wide	Specify system for data collection.
--cpu-mask=STRING	Specify which CPU(s) <string> to collect data on. For example, specify "2-8,10,12-14" to sample only CPUs 2



	through 8, 10, and 12 through 14.
-d SECONDS, --duration=SECONDS	Specify duration for the collection in seconds.
-l MILLISECONDS, --interval=MILLISECONDS	Specify an interval of data collection (for example, sampling) in milliseconds.
--app-working-dir=\$dir	Specify application working dir.
--target-install-dir=\$dir	Specify target installation folder.
-V, --version	Print version information.

➤ 分析报告 Report 相关:

参数	说明
--report=REPORT	Specify report type, example. --report summary or --report scheduling.
-r DIRECTORY, --result-dir=DIRECTORY	Specify the directory used for creating the data collection results, example ./r000ge or ./r000sch.

6.3 示例

6.3.1 服务器数据采集

Usage: malluma-analyzer [options] [--] <application> [<args>]

Example:

1. Collection data and report summary.

```
malluma-analyzer <--collect general-exploration> [-r ./r@@@ge] <a.out> [parm1 parm2 ]
malluma-analyzer <--collect general-exploration> [-r ./r@@@ge] <-p $pid> <-d 30>
malluma-analyzer <--collect general-exploration> [-r ./r@@@ge] <--system-wide> <-d 30>
malluma-analyzer <--collect general-exploration> [-r ./r@@@ge] <--system-wide> [-cpu-mask 2,5-8] <-d 30>
```

2. Report summary and print top-down total results to the screen.

```
malluma-analyzer <--report summary> <-r ./r@@@ge>
```

注: <>必选参数, []可选参数

核心采集脚本样例:				
malluma-analyzer	--collect	general-exploration	--	#采集应用 matrix.gcc
/home/duyanlin/matrix/linux/matrix.gcc				
malluma-analyzer	--collect	general-exploration	-- taskset -c 2-20	#采集应用 matrix.gcc, 并绑核 2-20
/home/duyanlin/matrix/linux/matrix.gcc				
malluma-analyzer	--collect	general-exploration	-r /home/test/r000ge --	#采集应用 matrix.gcc, 并指定采集结果路径
/home/duyanlin/matrix/linux/matrix.gcc				



malluma-analyzer --collect general-exploration -r /home/test/r000ge --app-working-dir /home/test -- /home/duyanlin/matrix/linux/matrix.gcc	#采集应用 matrix.gcc, 并指定应用执行路径
malluma-analyzer --collect general-exploration -p \$pid -d 5	#采集 PID, 5s
malluma-analyzer --collect general-exploration -p \$pid sleep 5	#采集 PID, 5s
malluma-analyzer --collect general-exploration --system-wide -d 5	#采集全系统, 5s
malluma-analyzer --collect general-exploration --system-wide sleep 5	#采集全系统, 5s
malluma-analyzer --collect general-exploration --system-wide /home/duyanlin/matrix/linux/matrix.gcc	#采集全系统, matrix.gcc 执行结束采集完成
malluma-analyzer --collect general-exploration --system-wide --cpu-mask 2,5-8 -d 5	#采集指定核, 5s
malluma-analyzer --collect general-exploration --system-wide --cpu-mask 2,5-8 sleep 5	#采集指定核, 5s
malluma-analyzer --report summary -r r000ge	#输出 summary 报告

7 FAQ

7.1 Websocket 服务中断

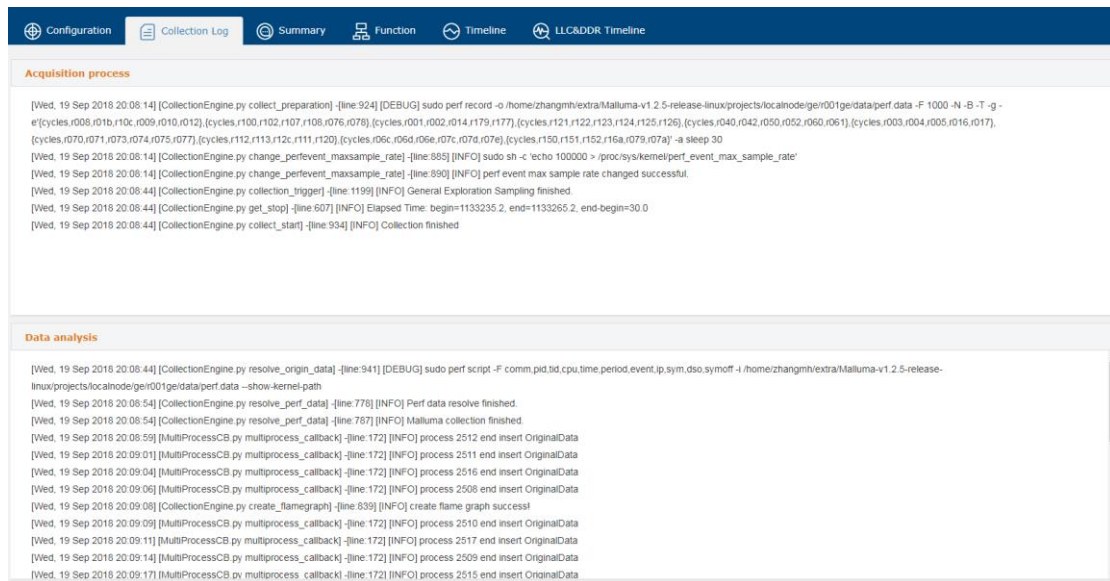
若部署 Malluma 环境因各种不同因素导致服务器重启, 会出现 websocket 中断, 需要手动重启 websocket 服务, 具体操作如下:

```
php Malluma/visualization/websocket/bin/mallumawebsocket start
```

备注: Malluma/visualization/websocket/bin/mallumawebsocket 此内容必须为 Malluma 部署的绝对路径。

7.2 Malluma 采集周期

关于针对全系统采集过程中配置的 Collection duration (s)值与实际采集过程时间的声明: Collection duration (s)仅指采集时间(即 Collection Log 的 Acquisition process 内容), 但是采集过程中采集数据成功后还需要进行数据解析和数据入库(即 Collection Log 的 Data analysis 内容)。例如某次采集配置的采集周期为 Collection duration (s)为 30(即采集 30s), 但是实际采集过程执行完成耗时为 15 分钟(该 15 分钟包含 30s 采集时间, 另外还包含采集数据解析和采集数据入库时间, 采集过程执行完成耗时与采集数据量大小有直接关系)。



7.3 Malluma 部署成功但不能访问 Web 端

若部署 Malluma 成功，且所有操作均检查正常，但 Web 端仍无法访问，建议关闭服务器环境的防火墙，并设置 SELinux 为关闭状态（Disabled）。

1. 关闭服务器防火墙：

`systemctl stop firewalld.service`

2. SELinux 查看和关闭相关操作如下：

（1）查看 SELinux 状态：

方法一： `/usr/sbin/sestatus -v` ##如果 SELinux status 参数为 enabled 即为开启状态

SELinux status: enabled

方法二： `getenforce` ##也可以用这个命令检查

（2）关闭 SELinux：

方法一：临时关闭（不用重启机器）：

`setenforce 0` ##setenforce 1 设置 SELinux 成为 enforcing 模式

方法二：修改配置文件需要重启机器：

`vim /etc/selinux/config` 文件

将 SELINUX=enforcing 改为 SELINUX=disabled

重启机器即可