**Расставляем сообщения Android Logcat на шкале времени VTune Amplifier**

**Введение**

     Известная пословица гласит: «Один ум хорошо, а два лучше». Она справедлива не только для житейских ситуаций, но и в делах отладки приложений. Объединяя «умы» различных программных средств, можно получить не только количественные, но и качественные преимущества. Например, союз **Android Logcat** и **VTune Amplifier** – это как раз такой случай.  
  
Android Logcat – это мощный отладочный инструмент. С его помощью можно получить массу полезной информации о системе или о приложениях с привязкой к временным меткам. Android предоставляет разработчикам стандартные API для логирования данных. Этими API удобно пользоваться при разработке и отладке приложений: достаточно добавить вызовы соответствующих команд в нужные места программного кода. Команды выводят сообщения в Logcat, где их можно просмотреть.  
  
VTune Amplifier для специализированных систем – это программный комплекс для профилирования. С его помощью можно анализировать и оптимизировать производительность операционных систем и приложений. VTune Amplifier имеет весьма полезное средство – панель шкалы времени. Оно позволяет разработчику просматривать сведения о производительности, привязанные ко времени.  
  
Как сопоставить сообщения из Logcat с временной шкалой VTune Amplifier? Это даст разработчикам возможность видеть сообщения из Logcat совместно со сведениями о производительности из временной шкалы VTune Amplifier. Как результат – можно будет узнать, что именно произошло в конкретное время и тут же понять, какую нагрузку на систему оказало то или иное событие, как оно повлияло на производительность.  
  
Intel VTune Amplifier способен обрабатывать и интегрировать в собственные наборы данных, полученные с помощью его встроенных инструментов, дополнительные сведения о производительности. Они могут быть получены с помощью внешних сборщиков данных или средствами тестируемого приложения. Для того чтобы добавить в VTune Amplifier внешние данные, нужен CSV-файл, обладающий определенной структурой, в котором находятся собранные сведения. Этот файл, для дальнейшей работы с ним, надо загрузить в VTune Amplifier.  
  
Подробности о создании собственных CSV-файлов с данными можно найти в руководстве пользователя VTune Amplifier. В частности, в разделе VTune Amplifier > User’s Guide > External Data Import есть сведения о том, как создавать такие файлы. Имеются там и примеры их внутренней структуры. Для того чтобы сообщения из Logcat оказались на временной шкале VTune, нужно, следуя руководству, преобразовать их в CSV-файл и загрузить полученный файл в VTune Amplifier.

**Пример**

Рассмотрим пример. Имеется приложение «com.example.Thread1». В нём есть функция, в которой выполняются тяжёлые вычисления. Вот как она, в общих чертах, выглядит:

|  |  |  |
| --- | --- | --- |
| 01 | void myfunction () | |
| 02 | { |

|  |  |  |
| --- | --- | --- |
| 03 | Log.v(“MYTEST”, “start block 1”); | |
| 04 | { |

|  |  |  |
| --- | --- | --- |
| 05 | ...//my computation block 1 | |
| 06 | } |

|  |  |  |
| --- | --- | --- |
| 07 | Log.v(“MYTEST”, “start block 2”); | |
| 08 | { |

|  |  |  |
| --- | --- | --- |
| 09 | ...//my computation block 2 | |
| 10 | } |

|  |  |  |
| --- | --- | --- |
| 11 | Log.v(“MYTEST”, “my computation finished”); | |
| 12 | } |

    Обратите внимание на то, что перед профилированием Android-приложения с помощью VTune Amplifier, приложение нужно подготовить. А именно, в его файле AndroidManifest.xml, в разделе*<application>*, должна быть включена возможность отладки:

Установить debug-версию приложения на Android-устройство можно, например, с помощью стандартных средств Android Studio. А после установки приложения Android Studio нужно будет закрыть, иначе VTune Amplifier не сможет собрать данные. Для того чтобы совместить сбор Logcat-сообщений и профилирование в VTune Amplifier, Logcat понадобится запустить из командной строки. Универсальное средство, подходящее и тем, кто работает в Windows, и тем – кто пользуется Linux – это оболочка bash.  
  
В Linux-системах bash присутствует по умолчанию. А вот если ваша хост-система – Windows, вам потребуется дополнительное ПО. Мы рекомендуем [Cygwin](https://www.cygwin.com/). Это приложение позволяет пользоваться bash-окружением в Windows. Подробности о работе с Logcat из командной строки можно узнать [здесь](http://developer.android.com/tools/debugging/debugging-log.html).  
  
По умолчанию, если мы, например, запустим Logcat командой «$ adb logcat -v threadtime», данные будут выводиться на экран. Гораздо удобнее перенаправить вывод во внешний файл. Делается это такой командой:

$ adb logcat -v threadtime > log.txt

Мы перенаправляем вывод от Logcat в файл log.txt, который будет помещен в текущей директории.  
Начинать сбор Logcat-данных нужно до запуска профилирования в VTune Amplifier, а останавливать (Ctrl + C в командной строке) – после.

Вернемся к нашему примеру. Судя по временной шкале VTune Amplifier, которая показана ниже, функция была выполнена 6 раз. ID главного потока приложения – 12271. Коричневые столбцы – это сведения о производительности (процессорное время), которые были собраны в ходе профилирования с помощью VTune Amplifier.

*Просмотр результатов анализа приложения на шкале времени VTune Amplifier*

1. При каждом выполнении функции, если собирать Logcat-сообщения командой вида «logcat -v threadtime», мы получим набор данных, показанный ниже.
2. 01-12 11:13:19.090 2174 2174 V MYTEST : start block 1
3. 01-12 11:13:19.260 2174 2174 V MYTEST : start block 2

01-12 11:13:19.500 2174 2174 V MYTEST : my computation finished

Получив эти данные, мы можем конвертировать их в CSV-файл подходящего формата, который подходит для загрузки в VTune Amplifier. В соответствии с документацией к VTune Amplifier, нужный нам CSV-файл может выглядеть так:

name,start\_tsc.UTC,end\_tsc,pid,tid

V/MYTEST : start block 1,2015-01-12 03:13:19.090,2015-01-12 03:13:19.090,2174,2174

V/MYTEST : start block 2,2015-01-12 03:13:19.260,2015-01-12 03:13:19.260,2174,2174

V/MYTEST:my computation finished,2015-01-12 03:13:19.500, 2015-01-12 03:13:19.500, 2174, 2174

Здесь, в качестве столбца данных «name», используется тег, который применялся при логировании, и строка сообщения. Время сообщения попадает в столбцы start\_tsc.UTC и end\_tsc. ID процесса и ID потока представлены в CSV-файле, соответственно, как pid и tid. Поля разделены запятыми.  
  
Теперь нужно перейти в VTune Amplifier и выполнить команду Analysis Type > Import from CSV, выбрав для импорта только что созданный файл с данными. Приложение загрузит данные и отобразит сообщения из Logcat на временной шкале со сведениями о производительности. На рисунке ниже показано, как перемещение мыши к маленькому жёлтому треугольнику, который указывает на точку расположения сообщения, выводит всплывающее окно с данными из Logcat, которые были сгенерированы нашим приложением.

*Просмотр сообщений из Logcat на шкале времени VTune Amplifier*

Вот несколько советов, которые касаются создания CSV-файлов.

1. Те отметки времени, которые отображаются в Logcat, берутся из Android OS. Они привязаны к часовому поясу, который выставлен на устройстве. Их, при создании CSV-файла, нужно поменять на отметки времени в стандарте UTC. Именно они нужны для успешного импорта данных в VTune Amplifier.
2. В имени CSV-файла должно быть указано имя хоста (hostname), на котором осуществляется сбор данных с помощью стороннего приложения. Например, имя файла может иметь такую структуру: [user-defined]-hostname-<hostname-of-system>.csv. Для целевых устройств, работающих под управлением Android, имя хоста есть в файле /proc/sys/kernel/hostname. Просмотреть его можно с помощью оболочки ADB.
3. Строковые данные, которые располагаются в столбце «name», можно настроить в соответствии с потребностями разработчика. В примере, рассмотренном выше, мы обошлись без изысков: в столбец «name» было помещено всё Logcat-сообщение. Обратите внимание на то, что в строковых данных, которые предназначены для столбца «name», не должно быть запятых. Знак запятой – это зарезервированный символ-разделитель, который не должен появляться в строках.
4. В формат CSV можно сконвертировать и сообщения ядра системы (например, от «dmesg»), импортировать в VTune Amplifier и просмотреть на временной шкале. Эта возможность весьма полезна для системных программистов, например, для тех, кто разрабатывает модули ядра или драйверы устройств. У сообщений из ядра системы ID процесса и потока должны быть установлены в 0. Обратите внимание и на то, что отсчёт отметок времени системных сообщений (например, от того же «dmesg») начинается с момента загрузки системы. Их нужно преобразовать так, чтобы они указывали на UTC-время вывода сообщений. Например, можно воспользоваться такой формулой расчёта временных меток сообщений: «Время сообщения» = «текущее время» — «время с начала работы системы» + «отметка времени сообщения». Под «текущим временем» здесь понимается текущее системное время. Его можно получить, воспользовавшись командой «date» в оболочке ADB. «Время с начала работы системы» — это время, которое прошло с начала загрузки ОС. Его можно найти в файле «/proc/uptime».

Вспомогательный скрипт

Мы создали экспериментальный bash-скрипт (logcat2vtune.sh), который позволяет упростить вышеописанные действия. Вы можете воспользоваться им для того, чтобы автоматизировать сбор Logcat-данных и создание CSV-файла. Скрипт может собирать сообщения, которые выводят в Logcat приложения, а так же – сообщения от ядра системы. Кроме того, он считывает сведения о целевой системе, разбирает результаты логирования и автоматически генерирует CSV-файлы.  
  
Для того чтобы воспользоваться этим скриптом, вам понадобится bash-окружение. Как было сказано выше, в Linux-системах это стандартная рабочая среда, а в Windows можно воспользоваться, например, Cygwin.  
Для того чтобы получить CSV-файл с данными в ходе профилирования с помощью VTune Amplifier, нужно сделать следующее:

1. Проверьте, доступен ли «adb» из командной строки. После этого запустите скрипт logcat2vtune.sh, указав необходимые ключи. Например, так: > ./logcat2vtune.sh -c logcat -g MYTEST
2. Запустите сбор данных о производительности в VTune Amplifier. Это можно сделать как из окна приложения, так и через командную строку.
3. Остановите сбор данных в VTune Amplifier.
4. Нажмите любую кнопку в окне командной строки, где работает logcat2vtune.sh, тем самым остановив сбор данных. Скрипт прочитает собранные данные логирования, разберет их с использованием регулярных выражений bash и сконвертирует в CSV-файл. Этот файл можно найти в текущей папке, с которой вы работаете из командной строки.
5. Загрузите полученный CSV-файл в VTune Amplifier. Теперь сообщения из Logcat можно просмотреть на временной шкале.

Вот несколько типичных примеров использования скрипта.

$logcat2vtune.sh -c logcat -g MYTEST

Эта команда позволяет собрать данные Logcat, отобрать сообщения со строкой «MYTEST» и сгенерировать CSV-файл. Данные Logcat скрипт собирает с помощью следующей команды:

$adb shell logcat -v threadtime

«MYTEST» – это строка, по которой осуществляется фильтрация сообщений из Logcat. Мы настоятельно рекомендуем пользоваться фильтрами, так как Logcat может выводить весьма длинные списки сообщений, в то время как разработчиков обычно интересуют лишь сообщения от их процесса. В качестве фильтра можно использовать имя тега Logcat-сообщений, ID процесса, ID потока или любую другую строку. Здесь можно использовать запятые, можно указывать несколько строк. Сообщения, в которых найдено совпадение с любой строкой, указанной при запуске скрипта с ключом «-g», будут отобраны и организованы в виде CSV-файла.

$logcat2vtune.sh -c dmesg -g MYDRIVER

С помощью такой команды можно собрать сообщения от «dmesg», отфильтровать их по строке «MYDRIVER» и создать CSV-файл.

$logcat2vtune.sh -c logcatk -g MYTEST,MYDRIVER

Эта команда позволяет собрать и обычные сообщения из Logcat, и сообщения от ядра, отфильтровать их по строкам «MYTEST» или «MYDRIVER» и создать CSV-файл. В результате на временной шкале VTune Amplifier можно будет просмотреть сообщения от обычных приложений пользовательского уровня и сообщения ядра от «dmesg». Эти данные скрипт собирает с помощью следующей команды:

>adb shell logcat -v threadtime -f /dev/kmsg | adb shell cat /proc/kmsg

В данном случае на шкале времени можно будет видеть сообщения от vmlinux с TID 0 и логи пользовательского уровня из Logcat с TID 1922.

Для того чтобы узнать подробности о работе со скриптом, воспользуйтесь следующей командой: «logcat2vtune.sh –h».

При желании, вы можете настроить скрипт под собственные нужды. Пожалуйста, учитывайте, что «logcat2vtune.sh» – это экспериментальный проект, он не прошёл всеобъемлющего тестирования. Если при работе с ним обнаружатся какие-либо проблемы – вы можете сообщить нам об этом.

Выводы

То, о чём мы здесь рассказали, применимо к продуктам Intel System Studio 2015 и Intel System Studio 2016. В материале освещена работа с хост-системами Windows (IA-32, Intel 64) и Linux (IA32, Intel 64). В качестве целевой системы рассмотрена ОС Android.

Воспользовавшись инструментами и приёмами работы, описанными выше, вы можете вывести профилирование и отладку Android-приложений на более высокий уровень, совместив информацию из Android Logcat с данными о производительности из VTune Amplifier. Надеемся, это поможет вашим приложениям покорить новые вершины производительности и экономичности.

|  |
| --- |
| [logcat2vtune.zip (2.92 KB)**Загрузить сейчас**](https://software.intel.com/sites/default/files/managed/b4/7f/logcat2vtune.zip) |
| [CorrelateAndroidLogcatMessagesWithVTuneTimeline.pdf (442.02 KB)**Загрузить сейчас**](https://software.intel.com/sites/default/files/managed/6d/f2/CorrelateAndroidLogcatMessagesWithVTuneTimeline.pdf) |

Для получения подробной информации о возможностях оптимизации компилятора обратитесь к нашему [Уведомлению об оптимизации](https://software.intel.com/ru-ru/articles/optimization-notice#opt-ru).