Підручник з KCachegrind

Перший автор документації: Josef Weidendorfer
Оновлення і виправлення: Federico Zenith
Переклад українською: Юрій Чорноіван

Підручник з KCachegrind

Зміст

1 Вступ **6**

1.1 Профілювання . . . 6
1.2 Способи профілювання . . . 6
1.3 Інструменти профілювання . . . 7
1.4 Візуалізація . . . 8

2 Користування KCachegrind **9**

2.1 Створення даних для візуалізації . . . 9
2.1.1 Callgrind . . . 9
2.1.2 OProfile . . . 9
2.2 Основи інтерфейсу користувача . . . 10

3 Основні концепції **11**

3.1 Модель даних для даних профілювання . . . 11
3.1.1 Елементи вартості . . . 11
3.1.2 Типи подій . . . 12
3.2 Стан візуалізації . . . 12
3.3 Частини графічного інтерфейсу користувача . . . 13
3.3.1 Бічні панелі . . . 13
3.3.2 Область перегляду . . . 13
3.3.3 Області у вкладках . . . 13
3.3.4 Синхронізований перегляд з вибором елемента у вкладці . . . 13
3.3.5 Синхронізація між вкладками . . . 13
3.3.6 Компонування . . . 13
3.4 Бічні панелі . . . 14
3.4.1 Плоский профіль . . . 14
3.4.2 Огляд частин . . . 14
3.4.3 Стек викликів . . . 14
3.5 Перегляди . . . 14
3.5.1 Тип події . . . 14
3.5.2 Списки викликів . . . 15
3.5.3 Карти . . . 15
3.5.4 Граф викликів . . . 15
3.5.5 Примітки . . . 15

4 Довідка щодо команд **16**

4.1 Головне вікно KCachegrind 16

4.1.1 Меню «Файл» 16

5 Запитання і відповіді **17**

6 Глосарій **18**

7 Подяки і ліцензія **20**

Анотація

KCachegrind — це інструмент для візуалізації даних профілювання, створений з використанням KDE Frameworks 5.

Розділ 1

Вступ

KCachegrind — програма, призначена для перегляду даних, створених інструментами для профілювання. У цій главі ми обговоримо призначення профілювання, способи його виконання, також наведемо приклади можливих інструментів профілювання.

1.1 Профілювання

Під час розробки програми одним з останніх кроків часто є оптимізація швидкодії. Оскільки оптимізація функцій, які використовуються рідко не має великого сенсу, потрібно знати кількість процесорного часу, яку споживає кожна з частин програми.

Для послідовного коду збирання статистичних даних щодо характеристик запущеної програми, зокрема час, витраченого на виконання функцій і рядків, зазвичай, достатньо. Збирання таких даних називається профілюванням. Програма запускається під керуванням інструменту профілювання, це надає резюме щодо виконання наприкінці. Навпаки, для розпаралеленого коду проблеми з швидкодією може бути спричинено очікуванням одного процесора на передавання даних від іншого процесора. Оскільки цей час очікування, зазвичай, важко пов'язати з кодом, у такому випадку краще створити трасування подій з часовими позначками. KCachegrind не може створити візуальний відповідник цього типу даних.

Після аналізу створених даних профілю ви маєте з легкістю виявити прогалини і вузькі місця у коді: наприклад, можна перевірити припущення щодо кількості викликів, виявлені діапазони коду може бути оптимізовано. Після цього ви можете перевірити результати оптимізації за допомогою повторного запуску профілювання.

1.2 Способи профілювання

Щоб точно виміряти час на виконання або записати події, які сталися під час виконання певного діапазону коду (наприклад функції), слід вставити додатковий код для вимірювання перед і після вказаного діапазону. Цей код зчитує час або загальну кількість подій, а потім обчислює різниці. Отже, перед виконанням початкового коду його слід змінити. Така зміна називається інструменталізацією. Інструменталізацію може виконати сам програміст, компілятор або система запуску. Оскільки цікаві діапазони коду зазвичай є вкладеними, накладання інструментів вимірювання завжди впливає на самі виміри. Таким чином, інструменталізацію слід виконувати вибірково, а результати слід сприймати з певною поправкою. Звичайно ж, це робити аналіз швидкодії точним виміром дуже складним процесом.

Точне вимірювання можливе, оскільки у сучасних процесорах передбачено апаратні лічильники (зокрема збільшення лічильників за один тік процесора), які збільшують значення кожного разу, коли трапляється подія. Оскільки нам потрібно пов'язати події з діапазонами коду, без лічильників нам би довелося обробляти кожну з подій шляхом збільшення значення лічильника для самого поточного діапазону коду. Виконання цього завдання у програмному

забезпеченні, звичайно ж, неможливе; але, за припущення, що розподіл подій у вихідному коді є подібним до того, який ми отримуємо під час аналізу кожної n-ої події замість кожної події, можна створити метод вимірювання з контрольованим рівнем помилки: такий метод називається вибірковим контролем. Заснований на часі вибірковий контроль (Time Based Sampling або TBS) використовує таймер для регулярного стеження за лічильником програм для створення гістограми коду програми. Заснований на подіях вибірковий контроль (Event Based Sampling або EBS) використовує апаратні лічильники сучасних процесорів, а також режим, за якого лічильник подій зменшує значення шляхом виклику інструменту переривання на час створення гістограми відповідного розподілу подій: у інструменті переривання лічильник подій завжди переініціалізується на значення n-ого кроку вибіркового контролю. Перевагою вибіркового контролю є те, що код змінювати не потрібно, але цей спосіб все ж проміжним: використане під час обґрунтування методу припущення буде точнішим, якщо n достатньо мале, але, чим меншим буде n, тим вищою буде кількість додаткових викликів інструменту переривання.

Ще одним методом вимірювання є імітація процесу, який відбувається у комп'ютерній системі під час виконання вказаного коду, тобто імітацією, заснованою на виконанні. Імітація завжди виконується на допомогою більшою чи меншою мірою точної моделі комп'ютера; але, якщо модель комп'ютера буде дуже детальною і даватиме дуже близькі до реальних дані, час імітації буде неприйнятно великим. Перевагою імітації є те, що ви можете додавати у вказаний код вимірювання/імітації довільної складності без впливу на результати. Виконання цього додавання перед виконанням (це називається інструменталізацію виконання) за допомогою початкового виконуваного файла є дуже зручним для користувача: не потрібне ніяке перезбирання. Імітація стає придатною для використання за імітації лише частин комп'ютера за допомогою простої моделі; іншою перевагою є те, що результати, створені за допомогою простих моделей часто простіше зрозуміти: часто проблеми, пов'язані зі справжнім апаратним забезпечення (обладнанням), є те, що у отриманих результатів буде включено ефекти, які є наслідком роботи різних частин комп'ютера.

1.3 Інструменти профілювання

Найвідомішим є інструмент профілювання GCC, gprof: для його використання вам потрібно зібрати програму з параметром `-pg`; після запуску програми буде створено файл `gmon.out`, який можна перетворити у придатну для читання форму за допомогою команди **gprof**. Одним з недоліків є потреба у перезбиранні для підготовки виконуваного файла, який має бути зв'язаним з бібліотеками статично. Тут використано створену компілятором інструменталізацію, — за якої буде виміряно час переходу викликів між функціями і відповідну кількість викликів з використанням TBS, — таким чином буде отримано гістограму розподілу часу у коді. З використанням обох цих джерел інформації можна виконати евристичне обчислення включеного часу функцій, тобто часу на обробку функції разом з усіма функціями, які було викликано з цієї функції.

Для точного вимірювання для подій, які сталися, існують бібліотеки з функціями, які можуть читати дані з апаратних лічильників швидкодії. Найвідомішим у цьому плані є латка PerfCtr для Linux® та архітектурно незалежні бібліотеки PAPI і PCL. Крім того, як ми про це говорили раніше, для точних вимірів потрібна інструменталізація коду. Можна використовувати або самі бібліотеки або автоматичні системи інструменталізації на зразок ADAPTOR (для інструменталізації вихідних кодів на FORTRAN) або DynaProf (вставка коду за допомогою DynInst).

OProfile — це загальносистемний інструмент для профілювання для Linux®, у якому використано опитування.

У багатьох аспектах зручним способом профілювання є використання Cachegrind або Callgrind, які є імітаторами, які використовують оболонку інструменталізації Valgrind. Оскільки за такої схеми немає потреби у доступі до апаратних лічильників (який є ускладненим у сучасних дистрибутивах Linux®), а виконувані файли, які профілюватимуться, залишаться незмінними, такий спосіб є непоганою альтернативою іншим інструментам профілювання. Наслідки недоліку імітації, — сповільнення, — можна зменшити імітацією лише цікавих частин програми або зменшенням кількості ітерацій під час циклу. Без інструменталізації вимірювання/імітації використання Valgrind коефіцієнт уповільнення змінюватиметься у діапазоні від 3 до 5. Крім того, якщо вас цікавлять лише граф викликів і кількість викликів, імітатор кешу можна вимкнути.

Імітація кешу є першим кроком у наближенні справжніх часів, оскільки у сучасних комп'ютерах виконання дуже чутливе до використання так званих *кешів* (маленьких і швидких буферів пришвидшує повторний доступ до тих самих комірок основної пам'яті). Cachegrind виконує імітацію кешу перехопленням доступу до пам'яті. Серед створеного потоку даних буде кількість доступів до інструкцій/даних у пам'яті і промахів кешу першого/другого рівнів, і співвідносить їх з рядками вихідних кодів і функціях запущеної програми. Комбінуванням цих кількостей промахів з використанням латентностей промахів від типових процесорів можна дістати оцінку витраченого часу.

Callgrind є розширенням Cachegrind, яке будує граф викликів програми на льоту, тобто граф, на якому буде показано спосіб, у який функції викликають одна одну, та кількість подій, які трапилися під час виконання функції. Крім того, дані профілю, які буде зібрано, може бути розподілено між потоками і контекстами ланцюжків викликів. Цей інструмент може надати дані профілювання на рівні інструкцій, що забезпечить коментування дизасембльованого коду.

1.4 Візуалізація

Типово, інструменти профілювання створюють значний об'єм даних. Бажання пересуватися вниз і вгору графом викликів, разом з швидким перемиканням режиму сортування функцій, і бачити різноманітні типи подій, викликає потребу у створенні програми з графічним інтерфейсом, призначеної для виконання цього завдання.

KCachegrind є інструментом візуалізації даних профілювання, який створено саме з вказаною вище метою. Незважаючи на те, що програму було від початку створено для перегляду даних з Cachegrind і Calltree, у ній передбачено інструменти перетворення, які призначено для показу даних профілювання, створених іншими інструментами. У додатку цього підручника ви зможете знайти опис формату файлів Cachegrind/Callgrind.

Окрім списку функцій, впорядкованого відповідно до вимірів включеної і виключної вартості, який за бажання можна згрупувати за файлами вихідних кодів, спільними бібліотеками або класами C++, у KCachegrind передбачено різноманітні режими візуалізації для вибраної функції, зокрема

- перегляд графу викликів, на якому буде показано частину графу викликів навколо вибраної функції,
- перегляд деревоподібної карти, за допомогою якого можна переглянути взаємозв'язки вкладених викликів разом з виміром включеної вартості для пришвидшення візуального виявлення проблемних функцій,
- перегляди вихідного коду і коментарів дизасемблера, за допомогою якого можна переглянути подробиці щодо пов'язаних з вартістю рядків вихідних кодів і інструкцій асемблера.

Розділ 2

Користування KCachegrind

2.1 Створення даних для візуалізації

Першим, у чому виникне потреба є створення даних щодо швидкодії шляхом вимірів характеристик виконання програми за допомогою інструменту профілювання. У самому KCachegrind не передбачено інструменту профілювання, але ця програма непогано пристосована для показу даних Callgrind або, за допомогою інструменту перетворення, даних OProfile. Хоча предметом цього підручника не є документування профілювання за допомогою цих інструментів, у наступному розділі наведено короткі початкові настанови щодо роботи з ними.

2.1.1 Callgrind

Callgrind є частиною Valgrind. Зауважте, що раніше програма називалася Calltree, але ця назва не відповідала призначенню програми.

Найпоширенішим способом використання є додавання перед командою запуску програми рядка `valgrind --tool=callgrind`, наприклад

```
valgrind --tool=callgrind myprogram myargs
```

Після переривання роботи програми буде створено файл `callgrind.out.pid`, який можна завантажити у KCachegrind.

Додатковою можливістю є створення дампів даних профілювання за виклику вказаної функції вашої програми. Наприклад, для того, щоб отримати дані профілювання для інструменту показу вебсторінки у Konqueror, вам може знадобитися створення дампу даних під час вибору пункту меню **Перегляд** → **Перезавантажити**. Цей пункт відповідає виклику `KonqMainWindow::slotReload`. Віддайте команду

```
valgrind tool=callgrind --dump-before=KonqMainWindow::slotReload konque
ror
```

Буде створено декілька файлів з даними для профілювання з додатковим послідовним номером наприкінці назви файла. Крім того, буде створено файл без такого номера наприкінці назви (його назва завершуватиметься PID процесу); після завантаження цього файла до KCachegrind, буде завантажено і всі інші файли, — їх можна переглянути у **огляді частин** і списку **частин**.

2.1.2 OProfile

OProfile можна отримати з його домашньої сторінки. Виконайте настанови зі встановлення з вебсайта; але, перш ніж виконувати ці настанови пошукайте відповідний пакунок у вашому дистрибутиві (наприклад SuSE®).

Загальносистемне профілювання можна виконувати лише користувачеві root. Оскільки за такого профілювання можна спостерігати за всіма діями у системі, слід виконувати наступні дії від імені користувача root. Спочатку, налаштуйте процес профілювання за допомогою графічного інтерфейсу **oprof_start** або інструменту командного рядка **opcontrol**. За стандартних налаштувань буде використано режим таймера (TBS, див. вступ). Щоб розпочати вимірювання, виконайте команду `opcontrol -s`. Після цього запустіть програму, яка вас цікавить, і виконайте команду `opcontrol -d`. Ця команда запише всі результати вимірювання до файлів у теці `/var/lib/oprofile/samples/` . Для того, щоб мати змогу візуалізувати дані у KCachegrind, виконайте таку команду у порожній теці:

```
opreport -gdf | op2callgrind
```

Ця команда призведе до створення багатьох файлів, по одному файлу на кожну з програм, які було запущено у системі. Кожен з цих файлів можна окремо завантажити до KCachegrind.

2.2 Основи інтерфейсу користувача

Після запуску KCachegrind з аргументом у вигляді назви файла даних профілювання або після завантаження цього файла за допомогою пункту меню **Файл** → **Відкрити...** ви побачите панель навігації, на якій міститиметься список функцій, розташовану ліворуч, праворуч же буде показано основну частину з даними візуалізації обраної функції. Цю область візуалізації можна налаштувати довільним чином так, щоб у ній було показано декілька візуалізацій одночасно.

Після першого запуску програми цю область буде поділено на верхню і нижню частини, у кожній з яких ви побачите декілька вкладок з панелями перегляду. Для пересування цими панелями скористайтеся контекстними меню вкладок. Ви також можете скоригувати розміри вкладок за допомогою панелей роздільників. Щоб пришвидшити перемикання між різними компонуваннями перегляду, скористайтеся пунктами меню **Перегляд** → **Розкладка** → **Перейти до наступного** (**Ctrl**+→) і **Перегляд** → **Розкладка** → **Перейти до попереднього** (**Ctrl**+←).

Для візуалізації є важливим вибір активного типу подій: для Callgrind це є, наприклад, промахи кешу або оцінка циклічності; для OProfile це є «Таймер» у найпростішому випадку. Ви можете змінити тип події за допомогою спадного списку на панелі інструментів або у перегляді **типу подій**. Перший огляд характеристик запуску можна отримати, якщо ви оберете функцію `main` у списку, розташованому ліворуч. Погляньте на візуалізацію графу виклику; там ви зможете побачити виклики, які було здійснено у вашій програмі. Зауважте, що у перегляді графу викликів буде показано лише функції з високими значеннями кількості подій. Подвійне клацання на позначках функцій на графі, перегляд буде змінено так, щоб було показано функції викликані вибраною вами функцією.

Щоб краще вивчити можливості графічного інтерфейсу програми на додачу до цього підручника, зазирніть до розділу документації на вебсайті. Крім того, кожен з віджетів KCachegrind має довідку «Що це?».

Розділ 3

Основні концепції

У цій главі наведено пояснення деяких концепцій KCachegrind та наведено визначення термінів, що вживаються у інтерфейсі програми.

3.1 Модель даних для даних профілювання

3.1.1 Елементи вартості

Лічильники вартості типів подій (на зразок промахів L2) пов'язано з елементами вартості, які є елементами відносно вихідних кодів або структур даних вказаної програми. Елементи вартості можуть бути не лише прості позиції у коді або даних, але і пари позицій. Наприклад, у виклику є джерело і призначення або адреса у даних може мати тип даних і позицію у коді, з якої було викликано її виділення.

Елементи вартості, відомі KCachegrind, може бути подано у наведеному нижче вигляді. Прості позиції:

Інструкція

Інструкція асемблера за заданою адресою.

Рядок коду у функції

Всі інструкції, які компілятор (за допомогою відомостей для усування вад) пов'язує з заданим рядком коду, який визначається за назвою файла і номером рядка і який виконується у контексті певної функції. Останній потрібен, оскільки рядок коду у вбудованій функції може з'являтися у контексті декількох функцій. Інструкції без жодних прив'язок до справжнього рядка коду пов'язуються з рядком з номером 0 у файлі `???`.

Функція

Всі рядки коду певної функції самі складаються з функцій. Кожна функція задається своєю назвою і розташування у певному бінарному об'єкті, якщо він доступний. Останній потрібен, оскільки бінарні об'єкти окремої програми можуть міститися функції з однаковими назвами (до цих функцій можна отримати доступ, наприклад за допомогою `dlopen` або `dlsym`; інструмент прив'язки під час виконання визначає функції у вказаному порядку пошуку використаних бінарних об'єктів). Якщо інструмент профілювання не може визначити назву символу функції, наприклад, оскільки інформація для усування вад недоступна, буде типово використано або адресу першої виконаної інструкції, або `???`.

Бінарний об'єкт

Всі функції, чий код перебуває у діапазоні вказаного бінарного об'єкта, основного виконуваного файла або спільної бібліотеки.

Файл вихідних кодів

Всі функції, перша інструкція яких пов'язується з рядком певного файла вихідних кодів.

Клас

Символічні назви функцій, типово, впорядковано за просторами назв, наприклад просторами назв C++ або класами у об'єктно-орієнтованих мовах; отже, клас може містити функції класу або самих вбудованих класів.

Частина профілювання

Певний часовий розділ запуску профілювання з певним ідентифікатором потоку, ідентифікатором процесу і рядком команди, яку слід виконати.

Як можна бачити зі списку, набір елементів вартості часто визначає інший елемент вартості; таким чином, існує послідовна ієрархія елементів вартості, структуру якої можна зрозуміти з наведеного вище опису.

Пари позицій:

- Виклик з адреси інструкції до функції призначення.
- Виклик з рядка вихідних кодів до функції призначення.
- Виклик з функції вихідних кодів до функції призначення.
- (Без)умовний перехід з вихідних кодів до інструкції призначення.
- (Без)умовний перехід з вихідних кодів до рядка призначення.

Переходи між функціями заборонено, оскільки такі переходи не мають сенсу у графі викликів; таким чином, конструкції на зразок обробки виключень і довгих переходів у C слід перетворити, за потреби, на виштовхування стеку викликів.

3.1.2 Типи подій

Довільні типи подій може бути визначено у даних профілювання наданням їм назви. Їх вартість відносно елемента вартості є 64-розрядним цілим числом.

Типи подій, чию вартість вказано у файлі даних профілювання, називаються справжніми подіями. Крім того, ви можете вказати формули для типів подій, обчислені на основі справжніх подій, які називаються успадкованими подіями.

3.2 Стан візуалізації

Стан візуалізації вікна KCachegrind включає:

- основні і вторинні типи подій, обраних для показу,
- групування функцій (використовується у списку **Функції профілювання** і розфарбування елементів),
- частини профілювання, чиї вартості слід включити до візуалізації,
- активний елемент вартості (наприклад вибраної на панелі функцій функції),
- вибраний елемент вартості.

Цей стан впливає на візуалізацію.

Вміст панелі перегляду завжди відповідатиме одному, активному, елементу вартості. Якщо певна панель не відповідає елементу вартості, її буде вимкнено (наприклад, якщо вибрати об'єкт ELF у списку групи, коментарі у вихідних кодах не матимуть сенсу).

Наприклад, для активної функції, у списку викликаних буде показано всі функції, які було викликано з активної функції: ви можете обрати одну з цих функцій, не роблячи її активною; крім того, якщо граф викликів показано поруч, у ньому також буде вибрано відповідну функцію.

3.3 Частини графічного інтерфейсу користувача

3.3.1 Бічні панелі

Бічні панелі є бічними вікнами, які можна розташовувати поряд з будь-якою з меж вікна KCachegrind. На них завжди міститься список елементів вартості, впорядкованих у той же спосіб.

- **Профіль функцій** є списком, у якому буде показано включену і виключну вартість, кількість викликів, назву і позицію функцій.
- **Огляд частин**
- **Стек викликів**

3.3.2 Область перегляду

Область перегляду, яка типово розташована у правій частині головного вікна KCachegrind, складається з одної (типової) або декількох вкладок, вирівняних горизонтально або вертикально. У кожному з переглядів вкладки містяться різноманітні дані для одного окремого елемента вартості. Назву цього елемента буде показано у верхній частині вкладки. Якщо відкрито декілька вкладок, лише одна з них буде активною. Назву елемента у активному перегляді вкладок буде показано жирним шрифтом, вона визначає активний елемент вартості вікна KCachegrind.

3.3.3 Області у вкладках

На кожній з вкладок може міститися до чотирьох областей перегляду, а саме верхня, права, ліва і нижня. У кожній з областей можуть міститися декілька переглядів візуалізації у вигляді стосу. Видиму частину області можна обрати за допомогою панелі вкладок. Панелі вкладок верхньої і правої областей знаходяться вгорі; панелі вкладок лівої і нижньої областей знаходяться внизу. Ви можете визначати різновид перегляду для кожної з областей за допомогою контекстних меню вкладок.

3.3.4 Синхронізований перегляд з вибором елемента у вкладці

Окрім активного елемента, у кожній вкладці є вибраний елемент. Оскільки у більшості з типів перегляду буде показано декілька елементів, активний елемент буде певним чином розташовано по центру, ви можете змінити вибраний елемент навігацією на панелі перегляду (клацанням кнопкою миші або за допомогою клавіатури). Типово, вибрані елементи буде підсвічено. Зміною вибраного елемента у одній з областей перегляду з вкладками, на всіх інших панелях у перегляді з вкладками буде відповідно підсвічено новий вибраний елемент.

3.3.5 Синхронізація між вкладками

Якщо буде відкрито декілька вкладок, зміна вибраного елемента на одній з вкладок призведе до зміни активації у наступному (праворуч/внизу) перегляді з вкладками. Цей тип прив'язки, наприклад, забезпечує швидку навігацію графами викликів.

3.3.6 Компонування

Компонування всіх переглядів з вкладками можна зберегти (за допомогою пункту меню **Перегляд** → **Розкладка**). Після дублювання поточного компонування (**Перегляд** → **Розкладка** → **Здублювати** (**Ctrl**++)) і зміни певних розмірів або пересування перегляду до іншої області перегляду з вкладками, ви можете швидко перемикатися між старим і новим компонуванням за допомогою комбінацій клавіш **Ctrl**+← і **Ctrl**+→. Набір компонувань буде збережено між сеансами роботи з KCachegrind над тією самою командою профілювання. Ви може зробити поточний набір компонувань як типовий для нових сеансів KCachegrind або відновити типовий набір компонувань.

3.4 Бічні панелі

3.4.1 Плоский профіль

Плоский профіль містить список груп і список функцій. У списку груп міститься перелік всіх груп, на які витрачаються ресурси системи, залежно від обраного типу груп. Список груп буде приховано, якщо групування буде вимкнено.

У списку функцій міститься перелік функцій вибраної групи (або всіх функцій, якщо групування вимкнено), впорядкованих за певним стовпчиком, наприклад включених або власних вартостей. Існує обмеження на максимальну кількість функцій, показаних у списку, його можна налаштувати за допомогою вікна, яке відкривається пунктом меню **Параметри → Налаштувати KCachegrind**.

3.4.2 Огляд частин

Під час виконання профілювання може бути створено декілька файлів даних профілювання, їх можна завантажити разом у KCachegrind. Ці файли буде показано на панелі **Огляд частин**, файли буде впорядковано у відповідності до часу створення; розміри прямокутників будуть пропорційними до вартості відповідних частин. Ви можете обрати одну або декілька частин, щоб обмежити перелік вартостей, які буде показано у інших переглядах KCachegrind, лише цими частинами.

Частини поділяються на менші частини. Існує режим розбиття і розділення включеної вартості:

Режим розбиття

> Ви побачите розбиття на групи для частини даних профілювання, відповідно до обраного типу груп. Наприклад, якщо обрано групи об'єктів ELF, ви побачите розфарбовані прямокутники для кожного використаного об'єкта ELF (спільної бібліотеки або виконуваного файла), їх розмір буде пропорційним до вартості використання ресурсів системи.

Режим діаграм

> Буде показано прямокутник, що відповідає включеній вартості поточної активної функції у частині. Цей перегляд, знову, буде розділено для того, щоб показати включені вартості викликаних елементів.

3.4.3 Стек викликів

Це цілком уявний «найбільш ймовірний» стек викликів. Його буде побудовано починаючи з поточної активної функції, туди буде додано виклики і викликані елементи з найвищою вартістю у верхню і нижню частину списку.

У стовпчиках **Вартість** і **Виклики** буде показано вартість, використану для всіх викликів з функції у рядку, розташованому вище.

3.5 Перегляди

3.5.1 Тип події

У списку **Тип подій** буде показано всі доступні типи вартості і відповідну власну і включену вартості поточної активної функції для відповідного типу події.

Вибором певного типу подій зі списку ви можете змінити тип вартостей, який буде показано у всіх інших переглядах KCachegrind, які буде обрано.

3.5.2 Списки викликів

Тут буде показано список викликів до/з поточної активної функції. За допомогою панелей **Всі виклики** і **Всі викликані функції** можна переглянути всі функції, яких можна досягти вздовж ланцюжка виклики/викликані, навіть якщо існують інші проміжні функції.

У переглядах списку викликів містяться:

- Безпосередні зовнішні виклики
- Безпосередні внутрішні виклики
- **Всі зовнішні виклики**
- **Всі внутрішні виклики**

3.5.3 Карти

Перегляд деревоподібної карти основного типу подій, вгору або вниз ієрархією викликів. Кожному розфарбованому прямокутнику відповідає функція, розмір прямокутника приблизно пропорційний до вартості у ресурсах системи виконання активної функції (крім того, існують певні обмеження малювання).

У разі показу **Карта викликів** на графі буде показано ієрархію всіх функцій, що викликали поточну активовану функцію. У разі показу **Карти викликаних** буде показано ієрархію всіх функцій викликаних активованою функцією.

Параметри вигляду можна знайти у контекстному меню. Щоб отримати точні пропорції, позначте пункт **Пропустити некоректні границі**. Оскільки побудова графів у подібному режимі є досить тривалою, ймовірно, вам слід обмежити максимальний рівень вкладеності до початку побудови. За вибору **Найкращого** режиму напрямок поділу для дочірніх елементів буде визначено зі співвідношення сторін батьківського елемента. За вибору режиму **Завжди найкращий** вибір відбуватиметься на основі доступного простору для елементів одного рівня. У режимі **Ігнорувати пропорції** прості для показу назви функції буде забрано перед малюванням дочірніх елементів. Зауважте, що пропорції за розмірами можуть бути дуже неточними.

Ви можете скористатися клавішами зі стрілками ліворуч/праворуч для пересування між елементами одного рівня, клавіші зі стрілочками вгору/вниз можна використати для пересування між рівнями вкладеності. Клавіша **Enter** активує поточний елемент.

3.5.4 Граф викликів

У цьому перегляді буде показано граф викликів навколо активної функції. Показана вартість є лише вартістю, яку було витрачено під час виконання самої активної функції; тобто вартість, показану для функції `main()`, — якщо вона є видимою, — має бути тією самою, що і вартість активної функції, оскільки вона є частиною включеної вартості `main()`, витраченої під час виконання активної функції.

Для циклів сині стрілки викликів позначатимуть, що ви маєте справу з штучним викликом, доданим для правильного показу, цей виклик ніколи не виконується.

Якщо розмір графу перевищує площу віджета, у одному з кутів буде показано елемент загального перегляду. Існують подібні до карти викликів параметри візуалізації, вибрану функцію буде підсвічено.

3.5.5 Примітки

У коментованих списках вихідних кодів/кодів асемблера буде показано рядки вихідних кодів/дизасембльовані інструкції поточної активної функції, а також (власну) вартість на виконання коду рядка вихідних кодів/інструкції. Якщо це був виклик, у вихідні коди буде вставлено рядки з подробицями щодо виклику: (включену) вартість виклику, кількість виконаних викликів і призначення викликів.

Оберіть певний рядок відомостей про виклики, щоб активувати призначення виклику.

Розділ 4

Довідка щодо команд

4.1 Головне вікно KCachegrind

4.1.1 Меню «Файл»

Файл → Створити (Ctrl-N)

Відкрити порожнє вікно верхнього рівня, у яке ви можете завантажите дані профілювання. Насправді, цей пункт не дуже то і потрібен, оскільки за допомогою пункту меню **Файл → Відкрити...** ви зможете відкрити нове вікно верхнього рівня, якщо у поточному вікні вже показано певні дані.

Файл → Відкрити (Ctrl-O)

Відкрити діалогове вікно відкриття файла KDE, у якому ви зможете обрати файл даних профілювання, який слід завантажити. Якщо у поточному вікні верхнього рівня вже показано певні дані, програма відкриє нове вікно; якщо ви бажаєте додати дані профілювання у поточне вікно, скористайтеся пунктом меню **Файл → Додати...**.

Назви файлів даних профілювання зазвичай завершуються на `.pid.частина-ідентифікатор потоку`, де *частина* і *ідентифікатор потоку* є необов'язковими, а *pid* і *частина* використовуються для декількох файлів даних профілювання, що стосуються одного запуску програми. Якщо завантажити файл, назва якого завершується лише на `pid`, буде завантажено не лише файли даних для цього запуску, але і файли з додатковими суфіксами.

Якщо існують файли даних профілювання `cachegrind.out.123` і `cachegrind.out.123.1`, після завантаження першого з файлів другий файл буде завантажено автоматично.

Файл → Додати...

Додати файл даних профілювання у поточне вікно. За допомогою цього пункту ви можете примусити програму до завантаження декількох файлів даних до одного вікна верхнього рівня, навіть якщо вони не належать до одного запуску програми відповідно до угоди щодо назв файлів даних профілювання. Цим можна, наприклад, скористатися для порівняльного аналізу.

Файл → Перезавантажити (F5)

Перезавантажити дані профілювання. Таке перезавантаження найцікавіше після того, як було створено інший файл даних профілювання для вже завантаженого запуску програми.

Файл → Вийти (Ctrl+Q)

Завершує роботу KCachegrind

Розділ 5

Запитання і відповіді

1. *Для чого призначено KCachegrind? Не можу здогадатися.*

   KCachegrind корисний на останніх стадіях розробки програмного забезпечення, які називаються профілюванням. Якщо ви не є розробником програм, вам не потрібен KCachegrind.

2. *У чому різниця між «Вкл.» і «Власна»?*

   Це атрибути вартості для функцій, що стосуються подій певного типу. Оскільки функції можуть викликати одна одну, має сенс відрізняти вартість самої функції («власну вартість») і вартість разом з всіма викликаними функціями («включену вартість»). «Власну» вартість іноді також називають «виключною».

   Отже, наприклад, для `main()` ви завжди матимете включену вартість близьку до 100%, тоді як власна вартість цієї функції є незначною, оскільки справжню роботу виконує інша функція.

3. *Якщо навести вказівник на позначку функції у перегляді* ***графу викликів*** *і двічі клацнути лівою кнопкою миші, для функції* ***`main()`*** *буде показано ту саму вартість вибраної функції. Хіба вартість всіх функцій рівна сталій 100% ?*

   Якщо ви активуєте вкладену функцію `main()` її вартість буде меншою за вартість `main()`. Для будь-якої функції, буде показано лише частину повної вартості функції, цю вартість було витрачено на виконання *активованої* функції; тобто, вартість, показана для будь-якої функції ніколи не може перевищувати вартість активованої функції.

Розділ 6

Глосарій

Елемент вартості

Абстрактний елемент, пов'язаний з вихідними кодами, з якими можна пов'язати кількість подій. Вимірами для елементів вартості є адреси коду (наприклад рядок вихідних кодів, функція), адреси даних (наприклад тип даних для доступу, об'єкт даних), адреса виконання (наприклад потік, процес) і пари або трійки вище наведених позицій (наприклад виклики, доступ до даних з інструкції, витискання даних з кешу).

Вартість подій

Сума вартостей подій певного типу, які трапляються під час виконання, пов'язаного з певним елементом вартості. Вартість буде прив'язано до елемента.

Тип події

Різновид події, з яким можна пов'язати елемент вартості. Існують справжні типи подій і успадковані типи подій.

Успадкований тип подій

Віртуальний тип подій, видимий лише на панелях перегляду, визначений формулою, за якою можна обчислювати дані на основі справжніх типів подій.

Файл даних профілювання

Файл, у якому містяться дані, отримані під час випробування з профілюванням (або його частини), або дані, отримані на основі обробки траси. Зазвичай, розмір цього файла пропорційний розмірів коду програми.

Частина «Дані профілювання»

Дані з файла даних профілювання.

Експеримент з профілювання

Програму буде запущено під наглядом інструменту профілювання, що призведе до можливого створення декількох файлів профілювання запущених частин або потоків.

Проєкт з профілювання

Налаштування для експериментів з профілювання, які використовуватимуться для одної з профільованих програм, можливо, у декількох версіях. Порівняння даних профілювання, типово, має сенс лише між даними профілювання, створеними під час експериментів одного проєкту профілювання.

Профілювання

Процес збирання статистичних відомостей щодо характеристик виконання запущеної програми.

Справжній тип подій

Тип подій, які можна виміряти інструментом. Для вказаного типу подій потрібне існування сенсора.

Траса

Послідовність подій з позначками часу, які сталися під час трасування запущеної програми. Зазвичай, розмір траси пропорційний до часу виконання запущеної програми.

Частина «Трасування»

див. "Частина «Дані профілювання»".

Трасування

Процес нагляду за виконанням програми зі збереженням списку подій програми, впорядкованого за часом у файлі виводу, трасі (Trace).

Розділ 7

Подяки і ліцензія

Дякуємо Julian Seward за його чудовий інструмент Valgrind, а також Nicholas Nethercote за додавання Cachegrind. Без цих програм KCachegrind ніколи не було б створено. Певні ідеї цього графічного інтерфейсу користувача також було запозичено з цих програм.

Крім того, спасибі на повідомлення про вади та пропозиції іншим користувачам.

Переклад українською: Юрій Чорноіван yurchor@ukr.net