O Manual do KCachegrind

Autor original da documentação: Josef Weidendorfer
Actualizações e correcções: Federico Zenith
Tradução: José Pires

Conteúdo

Resumo

O KCachegrind é uma ferramenta de visualização de dados de análise ('profiling') criada com o KDE Frameworks 5.

Capítulo 1

Introdução

O KCachegrind é um navegador para os dados produzidos pelas ferramentas de análise ('profiling'). Este capítulo explica para que é que serve a análise, como é que é feita e dá alguns exemplos das ferramentas de análise disponíveis.

1.1 Análise

Quando você desenvolve um programa, normalmente uma das últimas tarefas envolve as optimizações de performance. Dado que não faz sentido optimizar funções que são raramente utilizadas, convém você saber em que parte do seu programa a maioria do tempo é despendido.

Para o código sequencial, a recolha de dados estatísticos das características de execução dos programas, como os valores dos tempos despendidos nas funções e linhas de código é normalmente o suficiente. Isto é chamado normalmente de Análise ou 'Profiling'. O programa é executado sob o controlo de uma ferramenta de análise que fornece o resumo de uma execução no fim. Em contraste, para o código paralelo, os problemas de performance são normalmente causados quando um processador fica à espera dos dados do outro. Dado que este tempo de espera normalmente é atribuído de forma simples, aqui será melhor gerar traceamentos dos eventos com tempos marcados. O KCachegrind não consegue visualizar este tipo de dados.

Depois de analisar os dados produzidos, deverá ser fácil ver os pontos fortes e os críticos em termos de performance do código: por exemplo, podem ser tiradas conclusões sobre a quantidade de chamadas e as regiões de código que poderão ser optimizadas. No fim, o sucesso da optimização deverá ser verificado com uma nova análise.

1.2 Métodos de Análise

Uma medida exacta do tempo que passou ou dos eventos que ocorrem durante a execução de uma determinada região de código (isto é uma função) necessita da introdução de algum código de medida adicional, antes e depois da região indicada. Este código lê o tempo ou uma contagem de eventos global, calculando as diferenças. Assim, o código original terá de ser alterado antes da execução. Isto é chamado de instrumentação. Esta poderá ser criada pelo próprio programador, pelo compilador ou pelo sistema de execução. Dado que as regiões interessantes estão normalmente encadeadas, a sobrecarga da instrumentação influencia sempre a medida em si. Como tal, a instrumentação deverá ser feita de forma selectiva e os resultados terão de ser interpretados com cuidado. Obviamente, isto torna a análise de performance por medida exacta um processo bastante complexo.

É possível uma medida exacta devido aos contadores por 'hardware' (que incluem os contadores que incrementam a cada impulso de relógio) que vêm nos processadores modernos, os quais

são incrementados sempre que ocorre um evento. Dado que se pretende atribuir os eventos a regiões de código, sem os contadores, teria de se lidar com todos os eventos, incrementando um contador para a região de código em si. Fazer isso por 'software', obviamente, não é possível. Mas, assumindo que a distribuição de eventos pelo código-fonte é semelhante a procurar apenas por cada n-ésimo evento em vez de todos os eventos, foi criado um método de medida que é ajustado de acordo com a sobrecarga. É chamado de Amostragem. A Amostragem ao Longo do Tempo (TBS) usa um temporizador para ver regularmente o contador do programa para criar um histograma sobre o código do mesmo. A Amostragem Baseada em Eventos (EBS) tira partido dos contadores por 'hardware' dos processadores modernos e usa um modo em que é chamada uma rotina de tratamento de interrupções no caso de se atingir o valor mínimo de um contador, gerando um histograma, da distribuição do evento correspondente. Na rotina de tratamento, o contador do evento é sempre reinicializado para o 'n' do método de amostragem. A vantagem da amostragem é que o código não tem de ser alterado, mas é à mesma um compromisso: a hipótese anterior será mais correcta se o n for baixo, mas quanto mais baixo for o n, maior será a sobrecarga da rotina de tratamento da interrupção.

Outro método de medida é a simulação das coisas que ocorrem no sistema do computador enquanto executa um dado código, isto é uma simulação orientada pela execução. Obviamente, a simulação deriva sempre de um modelo de 'hardware' mais ou menos preciso. Para os modelos muito detalhados que se aproximam da realidade, o tempo de simulação poderá alto, de forma inaceitável para ser posto em prática. A vantagem é que o código de simulação/medida arbitrariamente complexo poderá ser introduzido num dado código sem perturbar os resultados. Se fizer isto directamente antes da execução (chamado de instrumentação durante a execução), usando o binário original, é bastante confortável para o utilizador. O método torna-se inútil quando se simula apenas partes de uma máquina com um modelo simples. Para além disso, os resultados produzidos pelos modelos simples são normalmente muito mais fáceis de compreender: o problema frequente com o 'hardware' real é que os resultados incluem efeitos sobrepostos de diferentes partes do sistema.

1.3 Ferramentas de Análise

A ferramenta de análise mais conhecida é o gprof do GCC: Você precisa de compilar o seu programa com a opção `-pg`; se correr o programa, irá gerar um ficheiro `gmon.out`, o qual poderá ser transformado num formato legível com o **gprof**. A desvantagem é o passo de compilação necessário para um dado executável, o qual terá de ser compilado estaticamente. O método aqui usado é a instrumentação gerada pelo compilador, que consiste na medida dos arcos de chamadas entre funções, bem como contadores para as chamadas respectivas, em conjunto com o TBS, o qual lhe dá um histograma com a distribuição do tempo pelo código. Usando ambos os dados, é possível calcular de forma heurística o tempo inclusivo das funções, isto é o tempo despendido numa função, em conjunto com todas as funções chamadas a partir dela.

Para uma medida exacta dos eventos que ocorrem, existem algumas bibliotecas com funções capazes de ler os contadores de performance do 'hardware'. As mais conhecidas são a actualização PerfCtr para o Linux®, e as bibliotecas independentes da arquitectura PAPI e PCL. De qualquer forma, uma medida exacta necessita de instrumentação no código, como é dito acima. Qualquer uma delas usa as próprias bibliotecas ou usa os sistemas de instrumentação automáticos como o ADAPTOR (para a instrumentação do código em FORTRAN) ou o DynaProf (injecção de código com o DynInst).

O OProfile é uma ferramenta de análise ao nível do sistema para Linux® que usa a amostragem.

Em vários aspectos, uma forma confortável de fazer uma Análise é com o Cachegrind ou o Callgrind, os quais são simuladores que usam a plataforma de instrumentação Valgrind durante a execução. Dado que não existe necessidade de aceder aos contadores do 'hardware' (o que é normalmente difícil com as instalações de Linux® de hoje em dia) e os binários a serem analisados podem ser deixados sem modificações, é uma boa forma alternativa para as outras ferramentas de análise. A desvantagem da lentidão da simulação poderá ser reduzida se fizer a simulação apenas nas partes interessantes do programa e, talvez, só apenas em algumas iterações de um

ciclo. Sem a instrumentação de medida/simulação, a utilização do Valgrind só terá um atraso numa gama de 3-5. Para além disso, quando apenas o grafo de chamadas e as contagens de chamadas forem de interesse, o simulador da 'cache' poderá ser desligado.

A simulação da 'cache' é o primeiro passo na aproximação dos tempos reais; como nos sistemas modernos, a execução é bastante sensível à exploração das *caches* que são pequenos e rápidos tampões de dados que aceleram os acessos repetidos às mesmas células da memória principal. O Cachegrind faz a simulação da 'cache', interceptando os acessos a memória. Os dados produzidos incluem o número de acessos à memória para dados e instruções, as falhas da 'cache' de 1º/2º nível e relaciona esses dados com as linhas de código e as funções do programa. Combinando estes valores e usando as latências de falhas típicas, é possível indicar uma estimativa do tempo despendido.

O Callgrind é uma extensão do Cachegrind que constrói o grafo de chamadas de um programa na altura, isto é como as funções se chamam umas às outras e quantos eventos acontecem enquanto uma função é executada. Para além disso, os dados da análise a serem recolhidos podem ser separados por tarefas ('threads') e por contextos de chamadas. Pode fornecer dados de análise ao nível da instrução para permitir a anotação do código descodificado.

1.4 Visualização

As ferramentas de análise produzem tipicamente uma grande quantidade de dados. A vontade de navegar facilmente para baixo e para cima no grafo de chamadas, em conjunto com uma alteração rápida do modo de ordenação das funções e a apresentação dos diferentes tipos de eventos, serve de motivo para criar uma aplicação GUI que desempenhe esta tarefa.

O KCachegrind é uma visualização para os dados de análise que preenche estes requisitos. Apesar de ser programada em primeiro lugar a partir da navegação dos dados do Cachegrind com a Calltree em mente, existem conversores disponíveis para apresentar os dados de análise produzidos pelas outras ferramentas. No apêndice, é dada uma descrição do formato do ficheiro do Cachegrind/Callgrind.

Para além de uma lista de funções ordenadas de acordo com as métricas de custo exclusivas ou inclusivas e, opcionalmente, agrupadas por ficheiros de código, bibliotecas partilhadas ou classes de C++, o KCachegrind contém diversas vistas para uma dada função, nomeadamente:

- um grafo de chamadas que mostra uma secção do grafo de chamadas em torno da função seleccionada,
- uma árvore que permite visualizar a relação de chamadas encadeadas, em conjunto com as métricas de custo inclusivas para uma detecção visual rápida das funções problemáticas,
- janelas do código-fonte e de anotação do código convertido para Assembly, permitindo ver os detalhes do custo relacionados com as linhas de código e as instruções de baixo-nível.

Capítulo 2

Usar o KCachegrind

2.1 Gerar Dados a Visualizar

Primeiro, uma pessoa deseja gerar os dados de performance, medindo aspectos das características de execução de uma aplicação, usando uma ferramenta de análise. O KCachegrind em si não inclui nenhuma ferramenta de análise, mas é bom a ser usado em conjunto com o Callgrind e, usando um conversor, também poderá ser usado para visualizar os dados produzidos com o OProfile. Apesar de o âmbito deste manual não ser a documentação da análise com estas ferramentas, a secção seguinte fornece vários tutoriais introdutórios para o ajudar a começar.

2.1.1 Callgrind

O Callgrind está disponível em Valgrind. Convém referir que se chamava anteriormente Calltree, mas esse nome era enganador.

O uso mais comum é anteceder a linha de comandos para iniciar a sua aplicação com o valgrind --tool=callgrind, como por exemplo

valgrind --tool=callgrind programa argumentos

Quando o programa terminar, será gerado um ficheiro `callgrind.out.pid` e que poderá ser carregado no KCachegrind.

Uma utilização mais avançada será descarregar os dados de análise, sempre que uma dada função da sua aplicação é chamada. Por exemplo, com o **konqueror**, para ver os dados de análise de modo a gerar apenas uma página Web, você poderia optar por gerar os dados sempre que carregasse no item do menu **Ver** → **Recarregar**. Isto corresponde a uma chamada ao `KonqMainWindow::slotReload`. Use o

```
valgrind --tool=callgrind --dump-before=KonqMainWindow::slotReloadkonqueror
```

. Isto irá produzir vários ficheiros de dados de análise com um número sequencial no fim do nome do ficheiro. Um ficheiro sem esse número no fim (terminando apenas no PID do processo) será também produzido. Se carregar este ficheiro no KCachegrind, todos os outros serão também carregados e poderão ser vistos na **Introdução das Partes** e na lista das **Partes**.

2.1.2 OProfile

O OProfile está disponível na sua página Web. Siga as instruções de instalação na página Web. Mas, antes disso, verifique se a sua distribuição não o oferece já como um pacote (como a SuSE®).

A análise ao nível do sistema só é permitida para o utilizador 'root', dado que todas as acções do sistema poderão ser observadas. Como tal, terão de ser feitas as seguintes acções como 'root'. Primeiro, configure o processo de análise, usando a GUI **oprof_start** ou a ferramenta da linha de comandos **opcontrol**. A configuração normal seria o modo de temporização (TBS, ver a introdução). Para iniciar a medida, execute o **`opcontrol-s`**. Depois execute a aplicação em que está interessado e, a seguir, invoque um **`opcontrol-d`**. Isto irá apresentar os resultados das medidas nos ficheiros sob a pasta `/var/lib/oprofile/samples/` . Para ser capaz de visualizar os dados no KCachegrind, execute numa pasta vazia:

```
opreport -gdf | op2callgrind
```

. Isto irá produzir uma quantidade de ficheiros, um por cada programa que estava a correr no sistema. Cada um deles poderá ser corrido no KCachegrind por si só.

2.2 Conceitos Básicos da Interface de Utilizador

Ao iniciar o KCachegrind com um ficheiro de dados de análise como argumento, ou depois de carregar um com a opção **Ficheiro** → **Abrir**, você irá ver uma barra lateral que contém a lista de funções à esquerda e a parte principal à direita, que consiste numa área com visualizações dos dados de uma função seleccionada. Esta área de visualização pode ser configurada de forma arbitrária para mostrar várias visualizações de uma vez.

À primeira vez, esta área estará dividida numa parte superior e outra inferior, tendo cada uma delas diferentes áreas que podem ser seleccionadas em páginas separadas. Para mover essas áreas, use o menu de contexto das páginas e ajustando as divisórias entre elas. Para mudar rapidamente de disposições de visualização, use as opções **Ver** → **Disposição** → **Ir para a Seguinte** (**Ctrl+**→) e **Ver** → **Disposição** → **Ir para a Anterior** (**Ctrl+**←).

Uma coisa importante para a visualização é o tipo de evento activo: para o Callgrind, este é isto é os 'Cache Misses' (Falhas na Cache) ou o Cycle Estimation (Estimativa da 'Cache') para o OProfile, este é o 'Temporizador' no caso mais simples. Você poderá alterar o tipo de evento com uma lista na barra de ferramentas ou na janela do Tipo de Evento. Uma primeira vista de olhos nas características de execução deverá ser apresentada quando você seleccionar a função main na lista da esquerda, e veja a visualização do grafo de chamadas. Aí, poderá ver as chamadas em curso no seu programa. Lembre-se que o grafo de chamadas só mostra as funções com uma grande quantidade de eventos. Se fizer duplo-click numa função do grafo, ela irá mudar para mostrar as funções chamadas pela seleccionada.

Para explorar mais a GUI, para além deste manual, dê uma vista de olhos na secção de documentação na página Web do projecto. Para além disso, cada elemento gráfico do KCachegrind tem ajudas 'O Que é Isto?'.

Capítulo 3

Conceitos Básicos

Este capítulo explica alguns conceitos do KCachegrind e introduz os termos usados na interface.

3.1 O Modelo dos Dados de Análise

3.1.1 Entidades de Custo

Os valores de custos dos tipos de eventos (como as Falhas de L2) são atribuídos às entidades de custo, as quais são itens relacionados com o código-fonte ou com as estruturas de dados de um dado programa. As entidades de custo podem ser não só posições no código ou nos dados, mas também tuplos de posição. Por exemplo, uma chamada tem uma origem e um destino, ou um endereço de dados poderá ter um tipo de dados e uma posição no código em que a sua alocação ocorreu.

As entidades de custo conhecidas pelo KCachegrind estão indicadas a seguir. Posições Simples:

- Instrução. Uma instrução de Assembly num endereço indicado.
- Linha de Código de uma Função. Todas as instruções que o compilador (através da informação de depuração) mapeia numa dada linha de código, identificada pelo nome do ficheiro de código e pelo número de linha, e que são executadas sob o contexto de uma dada função. A última é necessária, porque uma linha de código de uma função incorporada ('inline') poderá aparecer no contexto de várias funções. As instruções sem qualquer mapeamento numa linha de código são representadas pela linha 0 do ficheiro `???`.
- Função. Todas as linhas de código de uma dada função compõem a função em si. Uma função é identificada pelo seu nome e pela sua localização no ficheiro-objecto binário, se estiver disponível. A última é necessária porque os objectos binários de um único programa poderão conter funções com o mesmo nome (estas poderão ser acedidas, isto é, com o `dlopen`/`dlsym`; o editor de ligações durante a execução resolve as funções numa dada ordem de objectos binários). Se uma ferramenta de análise não conseguir detectar o nome do símbolo de uma função, isto é porque a informação de depuração não está disponível, tanto é usado o endereço da primeira instrução executada, ou então o `???`.
- Objecto Binário. Todas as funções cujo código esteja dentro do intervalo de um dado objecto binário, seja ele o executável principal ou uma biblioteca dinâmica.
- Ficheiro de Código. Todas as funções cuja primeira instrução esteja mapeada numa linha do ficheiro de código indicado.

- Classe. Os nomes dos símbolos das funções estão tipicamente ordenados de forma hierárquica em espaços de nomes, isto é os 'namespaces' de C++, ou as classes das linguagens orientadas por objectos. Como tal, uma classe poderá conter funções da classe ou outras classes embebidas nela.
- Parte de Análise. Alguma secção no tempo de uma execução da análise, com um dado ID de tarefa, ID de processo e uma linha de comandos executada.

Tal como é visto na lista, um conjunto de entidades de custo define normalmente outra entidade de custo. Como tal, existe uma hierarquia de inclusão das entidade de custo que deverá ser óbvia a partir da descrição acima.

Tuplos de posições:

- Uma chamada de uma instrução para uma função-alvo.
- Uma chamada de uma linha de código para uma função-alvo.
- Uma chamada de uma função de origem para uma função de destino.
- Um salto (in)condicional de uma instrução de origem para uma de destino.
- Um salto (in)condicional de uma linha de origem para uma de destino.

Os saltos entre funções não são permitidos, dado que isto não faz sentido num grafo de chamadas. Como tal, as sequências como o tratamento de excepções e os 'long jumps' do C terão de ser traduzidos em saltos na pilha de chamadas, de acordo com as necessidades.

3.1.2 Tipos de Evento

Podem ser indicados vários tipos de eventos arbitrários nos dados de análise, atribuindo-lhes um nome. O seu custo, relacionado com uma entidade de custo, é um inteiro de 64 bits.

Os tipos de eventos cujos tipos são indicados num ficheiro de dados de análise são chamados de eventos reais. Para além disso, uma pessoa poderá indicar fórmulas para os tipos de eventos, calculadas a partir dos eventos reais, chamadas de eventos inerentes.

3.2 Estado da Visualização

O estado da visualização de uma janela do KCachegrind inclui:

- o tipo primário e secundário dos eventos seleccionados para mostrar,
- o agrupamento de funções (usado na lista da Análise da Função e no colorir da entidade),
- as partes da análise cujos custos serão incluídos na visualização,
- uma entidade de custo activa (isto é uma função seleccionada a partir da barra de análise da função),
- uma entidade de custo seleccionada.

Este estado influencia as visualizações.

As visualizações são sempre apresentadas apenas para a entidade de custo activa de momento. Quando uma dada visualização não é apropriada para uma entidade de custo, fica desactivada: isto é, ao seleccionar um objecto ELF na lista de grupos através de um duplo-click, a anotação de código para um objecto ELF não faz sentido.

Por exemplo, para uma função activa, a lista de chamados mostra todas as funções chamadas a partir da função activa: um utilizador poderá seleccionar uma dessas funções sem a tornar activa. Se o grafo de chamadas é mostrado ao lado, irá seleccionar automaticamente a mesma função.

3.3 Partes da GUI

3.3.1 Barras Laterais

As barras laterais são janelas laterais que poderão ser colocadas em qualquer extremo de uma janela do KCachegrind. Elas contêm sempre uma lista das entidades de custo, ordenadas de uma determinada forma.

- A **Análise da Função** é uma lista de funções que mostram o custo inclusivo e o exclusivo, o número de chamadas, o nome e a posição das funções.
- **Introdução às Partes**
- **Pilha de Chamadas**

3.3.2 Área de Visualização

A área de visualização, tipicamente do lado direito da janela principal do KCachegrind, é composta por uma (a predefinida) ou mais páginas, quer alinhadas na horizontal quer na vertical. Cada página contém diferentes áreas de visualização com apenas uma entidade de custo de cada vez. O nome desta entidade é indicado no cimo da página. Se existirem várias páginas, só uma é que estará activa. O nome da entidade da página activa é mostrado a negrito e determina a entidade de custo activa da janela do KCachegrind.

3.3.3 Áreas de uma Página

Cada página poderá conter até quatro áreas de visualização, nomeadamente a de Topo, Direita, Esquerda e Fundo. Cada área poderá conter várias vistas empilhadas. A área visível é seleccionada por uma barra de páginas. As barras de páginas na área da direita e de topo estão em cima, enquanto que as barras de páginas da esquerda e de baixo estão no fundo. Você poderá indicar que tipo de visualização deverá ir para determinada área com o menu de contexto das páginas.

3.3.4 Visualização Sincronizada da Entidade Seleccionada numa Página

Para além de uma entidade activa, cada página tem uma entidade seleccionada. Como a maioria dos tipos de visualização mostram várias entidades com a activa centrada, você muda o item seleccionado se navegar dentro de uma visualização (carregando com o rato ou usando o teclado). Tipicamente, os itens seleccionados são mostrados de forma realçada. Se alterar a entidade seleccionada numa das visualizações de uma página, todas as outras visualizações da página irão ficar realçadas de igual forma na nova entidade seleccionada.

3.3.5 Sincronização entre Páginas

Se existirem várias páginas, uma mudança de selecção numa das páginas leva a uma mudança da activação na próxima página (à direita/em baixo). Este tipo de associação isto é deverá permitir uma navegação rápida nos grafos de chamadas.

3.3.6 Disposições

A disposição de todas as páginas de uma janela poderá ser gravada (veja o item do menu **Ver** → **Disposição**). Depois de duplicar a disposição actual (**Ver** → **Disposição** → **Duplicar** (**Ctrl++**)) e alterar alguns tamanhos ou mudar uma área de visualização de posição para outra área de uma página, você poderá mudar rapidamente entre a disposição antiga e a nova com as combinações **Ctrl**+← e **Ctrl**+→. O conjunto de disposições será guardado entre sessões do KCachegrind do mesmo comando analisado. Você poderá tornar o conjunto de disposições o predefinido para as novas sessões do KCachegrind ou reponha o conjunto de disposições por omissão.

3.4 Barras Laterais

3.4.1 Análise Simples

A **Análise Simples** contém uma lista de grupos e outra lista de funções. A lista de grupos contém todos os grupos em que o custo é despendido, dependendo do tipo de grupo escolhido. A lista de grupos fica escondida quando o agrupamento está desligado.

A lista de funções contém as funções do grupo seleccionado (ou todas as funções se o agrupamento estiver desligado), ordenadas por uma dada coluna, isto é os custos da própria ou os custos inclusos despendidos até então. Existe um número máximo de funções apresentado na lista que é configurável na opção **Configuração** → **Configurar o KCachegrind**.

3.4.2 Introdução às Partes

Na execução de uma análise, poderão ser produzidos vários ficheiros de dados de análise que poderão ser carregados em conjunto no KCachegrind. A barra de **Introdução das Partes** mostra estes ficheiros, ordenados na horizontal de acordo com a hora de criação; os tamanhos dos rectângulo são proporcionais ao custo despendido nas partes. Você poderá seleccionar uma ou várias partes para restringir os custos apresentados nas outras zonas do KCachegrind apenas para estas partes.

As partes são, por sua vez, sub-divididas num modo de partição e num modo repartido por custo inclusivo:

Modo de Partição

O utilizador vê a repartição em grupos para uma parte de dados de análise, de acordo com o tipo de grupo seleccionado. Por exemplos, se forem seleccionados os grupos de objectos ELF, você irá ver rectângulos coloridos para cada objecto ELF usado (biblioteca dinâmica ou executável), dimensionado de acordo com o custo nele despendido.

Modo de Diagrama

É mostrado um rectângulo com o custo inclusivo da função activa de momento na parte. Este, por sua vez, vai sendo repartido para mostrar os custos inclusos das funções chamadas por ela.

3.4.3 Pilha de Chamadas

Esta é uma pilha de chamadas 'mais prováveis' puramente fictícia. É criada a partir da função activa de momento e adiciona as funções chamadoras/chamadas com o maior custo no seu topo e no seu fundo.

As colunas **Custo** e **Chamadas** mostram o custo usado para todas as chamadas da função na linha acima.

3.5 Áreas

3.5.1 Tipo de Evento

A lista **Tipo de Evento** mostra os tipos de custos disponíveis e o custo correspondente à própria e o inclusivo para a função activa de momento, para esse tipo de evento.

Se escolher um tipo de evento na lista, você poderá alterar o tipo de custos apresentados em todo o KCachegrind, de modo a ser o tipo seleccionado.

3.5.2 Listas de Chamadas

Estas listas mostram as chamadas de/para a função activa de momento. Entende-se por **Todos os Chamadores** e **Todos os Chamados** as funções que poderão ser acedidas no sentido da chamadora/chamada, mesmo que existam outras funções pelo meio.

A lista de chamadas inclui:

- **Chamadores** Directos
- **Chamados** Directos
- **Todos os Chamadores**
- **Todos os Chamados**

3.5.3 Mapas

Uma visualização em árvore do tipo de evento primário, para cima ou para baixo, na hierarquia de chamadas. Cada rectângulo colorido representa uma função; o seu tamanho tenta ser proporcional ao custo despendido na função activa enquanto está a correr (contudo, existem restrições de desenho).

Para o **Mapa dos Chamadores**, o gráfico mostra a hierarquia encadeada de todas as funções que chamam a função activa de momento; no caso do **Mapa dos Chamados**, mostra a hierarquia respectiva, mas para as funções chamadas pela função activa.

As opções de aparência poderão ser acedidas no menu de contexto. Para obter proporções de tamanho exactas, escolha a opção **Esconder os contornos incorrectos**. Dado que este modo poderá ocupar bastante tempo, o utilizador poderá desejar limitar o nível máximo de encadeamento do desenho antes. O **Melhor** determina a direcção da repartição dos filhos, a partir das proporções do pai. O **Sempre o Melhor** decide sobre o espaço restante de cada elemento do mesmo nível. O **Ignorar as Proporções** ocupa o espaço para o nome da função, antes de desenhar os filhos. Lembre-se que as proporções podem ficar totalmente erradas.

A navegação com o teclado está disponível com as teclas de cursores esquerda/direita para navegar nos elementos do mesmo nível, enquanto que os cursores cima/baixo sobem/descem um nível de encadeamento. O **Enter** activa o item actual.

3.5.4 Grafo de Chamadas

Esta janela mostra o grafo de chamadas em torno da função activa. O custo apresentado é apenas o custo despendido enquanto a função estava de facto a correr; isto é, o custo mostrado para o `main()` - se for visível - deverá ser o mesmo que o custo da função activa, dado que faz parte do custo inclusivo do `main()` despendido enquanto a função activa estava em execução.

Para os ciclos, as setas de chamadas a azul indicam que esta é uma chamada artificial adicionada para desenhar correctamente o que, de facto, nunca ocorreu.

Se o grafo for maior que a área de desenho, é mostrada uma vista geral num dos lados. Existem opções de visualização semelhantes às da Árvore de Chamadas; a função seleccionada está realçada.

3.5.5 Anotações

As listas anotadas de código ou Assembly mostram as instruções de código ou descodificadas para Assembly da função activa de momento, em conjunto com o custo (da própria) despendido ao executar o código de uma linha de código ou instrução. Se ocorreu uma chamada, as linhas com os detalhes da chamada são introduzidas no código: o custo (inclusivo) despendido dentro da chamada, o número de chamadas que ocorreu e o destino da chamada.

Seleccione uma linha de informação da chamada para activar o destino da chamada.

Capítulo 4

Referência de Comandos

4.1 A janela principal do KCachegrind

4.1.1 O Menu Ficheiro

Ficheiro → Novo (Ctrl-N)

Abre uma janela de topo em branco para onde poderá carregar os dados de análise. Esta acção não é realmente necessária, dado que o **Ficheiro → Abrir** dar-lhe-á uma nova janela de topo, quando a actual já estiver a mostrar alguns dados.

Ficheiro → Abrir (Ctrl-O)

Abre a Janela de Abertura de Ficheiros do KDE para escolher o ficheiro de dados de análise a ser carregado. Se existirem já alguns dados visíveis na janela de topo actual, esta opção irá abrir uma nova janela. Se quiser abrir dados adicionais de análise na janela actual, use o **Ficheiro → Adicionar**.

O nome dos ficheiros de dados de análise normalmente termina em `.pid.part-idTarefa`, onde o *part* e o *idTarefa* são opcionais e; o *pid* e o *part* opcionais são usados para vários ficheiros de dados de análise que pertençam uma execução de uma aplicação. Se ler um ficheiro que termine apenas em `pid`, os ficheiros de dados eventualmente existentes para esta execução, mas sem terminações adicionais, são também carregados.

Exemplo: Se existirem os ficheiros de dados de análise `cachegrind.out.123` e `cachegrind.out.123.1`, ao carregar o primeiro, o segundo será também carregado automaticamente.

Ficheiro → Adicionar

Adiciona um ficheiro de dados de análise à janela actual. Com isto, você poderá obrigar vários ficheiros de dados a serem carregados para a mesma janela de topo, mesmo que não sejam da mesma execução, tal como está definido através da convenção de nomes dos ficheiros de dados de análise. Por exemplo, poderá ser usado para uma comparação 'lado-a-lado'.

Ficheiro → Recarregar (F5)

Volta a carregar os dados de análise. Isto é mais interessante, depois de outro ficheiro de dados de análise ter sido gerado para uma execução de uma aplicação já carregada.

Ficheiro → Sair (Ctrl-Q)

Sai do KCachegrind

Capítulo 5

Perguntas e Respostas

1. *Para que é que serve o KCachegrind? Não faço a mínima ideia.*

   O KCachegrind é útil numa fase posterior do desenvolvimento do 'software' que é a análise de performance ('profiling'). Se você não programar aplicações, não precisa do KCachegrind.

2. *Qual é a diferença entre o **Incl.** e o **Próprio**?*

   Estes são atributos de custos para as funções, no que respeita a um dado tipo de evento. Dado que as funções se podem chamar umas às outras, faz sentido distinguir o custo da função em si ('Custo da Própria') e o custo que inclui todas as funções chamadas ('Custo Inclusivo'). O 'Próprio' é referido também como custo 'Exclusivo'.

   Por isso, por exemplo para o `main()`, você irá ter sempre um custo inclusivo de quase 100%, enquanto que o custo da própria função é infinitesimal face ao real trabalho desempenhado nas outras funções.

3. *Se fizer duplo-click numa função qualquer do **Grafo de Chamadas**, ela mostra para a função `main()` o mesmo custo que para a função seleccionada. Não é suposto ser constante e igual a 100%?*

   Você activou uma função sob a `main()` com um custo menor que o da `main()`. Para qualquer função, só é apresentada essa parte do custo completo da função, sendo ela despendida enquanto a função *activa* está em execução, isto é, o custo mostrado para qualquer função nunca pode ser maior que o custo da função activada.

Capítulo 6

Glossario

Entidade de Custo

Um item abstracto relacionado com o código-fonte, para o qual poderão ser atribuídas as contagens de eventos. As dimensões das entidades de custo são a localização no código (isto é, linha de código, função), a localização dos dados (isto é tipo dos dados acedidos, o objecto de dados), a localização da execução (isto é, a tarefa ou processo) e os tuplos das posições acima indicadas (isto é, as chamadas, o acesso aos objectos pela instrução, os dados obtidos a partir da 'cache').

Custos do Evento

A soma dos eventos de um determinado tipo que ocorrem enquanto a execução está relacionada com uma dada entidade de custo. O custo em si é atribuído à entidade.

Tipo de Evento

O tipo de evento do qual os custos poderão ser atribuídos a uma entidade de custo. Existem os tipos de eventos reais e os inerentes.

Tipo de Evento Inerente

Um tipo de evento virtual que só aparece na visualização e que é definido por uma fórmula calculada a partir dos tipos de eventos reais.

Ficheiro de Dados de Análise

Um ficheiro que contém os dados medidos numa experiência de análise (ou parte dela) ou produzidos depois da análise de um traceamento. O seu tamanho é tipicamente linear com o tamanho do código do programa.

Componente de Dados de Análise

Dados de um ficheiro de dados de análise.

Experiência de Análise

Uma execução de um programa supervisionada por uma ferramenta de análise, gerando possivelmente vários ficheiros de dados das partes e/ou tarefas dessa execução.

Projecto de Análise

Uma configuração para as experiências de análise usada para um programa que tenha de ser analisado, talvez para várias versões. As comparações dos dados de análise só fará tipicamente sentido entre dados de análise produzidos num único projecto de análise.

Análise ('Profiling')

O processo de reunião de informação estatística sobre as características das execuções dos programas.

Tipo de Evento Real

Um tipo de evento que poderá ser medido por uma ferramenta. Necessita da existência de um sensor para o tipo de evento indicado.

Traceamento

Uma sequência de eventos ao longo do tempo que ocorreu durante a supervisão da execução de um programa. O seu tamanho é tipicamente linear com o tempo de execução do programa.

Componente de Traceamento

Ver "Componente de Dados de Análise".

Traceamento

O processo de supervisionar a execução de um programa e registar os eventos que ocorrem, ordenados por data e hora, num ficheiro de resultado, o ficheiro de Traceamento.

Capítulo 7

Créditos e Licença

Obrigado ao Julian Seward pelo seu excelente Valgrind, e ao Nicholas Nethercote pela adição do Cachegrind. Sem estes programas, o KCachegrind não existiria. Algumas das ideias para esta GUI foram dadas por eles, também.

Muito obrigado a todos os relatórios de erros e sugestões dos vários utilizadores.

Tradução de José Nuno Pires zepires@gmail.com