Manual do KCachegrind

Autor original da documentação: Josef Weidendorfer
Atualizações e correções: Federico Zenith
Tradução: Marcus Gama

Conteúdo

Resumo

O KCachegrind é uma ferramenta de visualização de dados de auditoria, escrito usando o ambiente KDE.

Capítulo 1

Introdução

O KCachegrind é um navegador para dados produzidos por ferramentas de auditoria. Este capítulo esclarece para que serve um auditoria, como ela é feita, e fornece alguns exemplos de ferramentas de auditoria disponíveis.

1.1 Auditoria

Quando você desenvolve um programa, um dos últimos passos muitas vezes envolve otimizações de desempenho. Como não faz sentido otimizar funções raramente usada, uma vez que elas não consumirão muito tempo de execução, você precisa saber em qual parte do seu programa a maior parte do tempo é consumido.

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

Após analisar os dados da auditoria produzida, deve ser fácil ver os trechos mais importantes e pontos de estrangulamento do código. Por exemplo, suposições sobre a contagem de chamadas podem ser verificadas, e identificadas regiões do código que pode ser otimizadas. Além disso, sucesso da otimização pode ser verificado novamente com a execução de outra auditoria.

1.2 Métodos de Auditoria

Uma medida exata do tempo que passou ou dos eventos que ocorrem durante a execução de uma determinada região de código (por exemplo uma função) necessita da inserção 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 que ser alterado antes da execução. Isto é chamado de instrumentação. Ela poderá ser criada pelo próprio programador, pelo compilador ou pelo sistema de execução. Uma vez que as regiões interessantes estão normalmente encadeadas, a sobrecarga da instrumentação influencia sempre a medida em si. Assim, a instrumentação deverá ser feita de forma seletiva e os resultados terão que ser interpretados com cuidado. Obviamente, isto torna a análise de performance por medida exata um processo bastante complexo.

A medição exata é possível em função os contadores de hardware (incluindo contadores incrementados em um tique de tempo) fornecidos nos processadores modernos, que são incrementados sempre que um evento acontece. Como nós desejamos atribuir eventos para regiões de código, sem os contadores, nós teríamos que manipular cada evento incrementando um contador para a região de código atual por nós mesmos. Fazer isto no software é claramente impossível; mas na suposição que a distribuição do evento pelo código-fonte é semelhante quando procuramos somente a cada enésimo evento ao invés de evento por evento, nós construímos um método de medição que é ajustável em função do todo. Isto é chamado Amostragem. A Amostragem Baseada em Tempo (do inglês TBS) é feita usando um temporizador que regularmente verifica o contador do programa para criar um histograma ao longo do código do programa. Amostragem Baseada em Evento (do inglês EBS) é feita aproveitando os contadores de hardware dos modernos processadores, e é feita usando um modo onde um manipulador de interrupção é chamado em um contador abaixo do fluxo, gerando um histograma da distribuição do evento correspondente. No manipulador, o contador do evento está sempre reinicializado para o n do método de amostragem. A vantagem da amostragem é que o código não precisa ser mudado, mas ela ainda deve atentar para uma restrição: a suposição acima será mais correta se n for pequeno, mas quanto menor o n, maior o todo do manipulador de interrupção.

Outro método de medida é a simulação das coisas que ocorrem no sistema do computador enquanto executa um determinado 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 ser alto, de forma inaceitável para ser posto em prática. A vantagem é que o código de simulação/medida arbitrariamente complexo poderá ser introduzido num determinado código sem perturbar os resultados. Fazer isto diretamente antes da execução (chamado de instrumentação durante a execução), usando o binário original, é bastante confortável para o usuário. O método torna-se inútil quando se simula apenas partes de uma máquina com um modelo simples. 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 Auditoria

A ferramenta de análise mais conhecida é o gprof do GCC: você precisa compilar o seu programa com a opção `-pg`; rodar o programa, irá gerar um arquivo `gmon.out`, que poderá ser transformado num formato legível com o **gprof**. A desvantagem é o passo de compilação necessário para preparar o executável, que terá que 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 respectivas chamadas, juntamente com o TBS, que lhe fornece 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 dispendido numa função, juntamente com todas as funções chamadas a partir dela.

Para uma medida exata dos eventos que ocorrem, existem algumas bibliotecas com funções capazes de ler os contadores de performance do 'hardware'. As mais conhecidas são a atualização PerfCtr para o Linux®, e as bibliotecas independentes da arquitetura PAPI e PCL. De qualquer forma, uma medida exata necessita de instrumentação no código, como dito anteriormente. 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 (injeção de código com o DynInst).

O OProfile é uma ferramenta de análise a 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, que são simuladores que usam a plataforma de instrumentação Valgrind durante a execução. Uma vez que não existe necessidade de acessar os 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 fazendo 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 num fator de 3 à 5. Além disso, quando apenas o gráfico 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 buffers 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 dispendido.

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

1.4 Visualização

As ferramentas de análise produzem tipicamente uma grande quantidade de dados. O desejo de navegar facilmente para baixo e para cima no gráfico 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 um aplicativo GUI que desempenhe esta tarefa.

O KCachegrind é uma ferramenta de 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 arquivo do Cachegrind/Callgrind.

Além de uma lista de funções ordenadas de acordo com medidas de custo exclusivas ou inclusivas, e opcionalmente agrupadas por arquivo fonte, biblioteca compartilhada ou classe C++, os recursos do KCachegrind incluem diversas formas de visualização para uma função selecionada, quais sejam:

- uma visão de gráfico de chamadas, que mostra uma seção do gráfico de chamadas ao redor da função selecionada,
- uma visão de mapa em árvore, que permite visualizar chamadas aninhadas relacionadas juntas com métrica de custo inclusiva para rápida detecção visual de funções problemáticas,
- código-fonte e visões anotadas desassembladas, permitindo ver detalhes do custo relacionado às linhas de código e instruções assembler.

Capítulo 2

Usando o KCachegrind

2.1 Gerando Dados para Visualizar

Primeiro, devemos gerar dados de desempenho medindo os aspectos das características em tempo de execução de um aplicativo, usando uma ferramenta de auditoria. O KCachegrind propriamente dito não inclui nenhuma ferramenta de auditoria, mas pode ser melhor aproveitado em conjunto com o Callgrind, e usando um conversor, pode também ser usado para visualizar dados produzidos com o OProfile. Apesar do escopo deste manual não ser o de documentar como auditar com estas ferramentas, a próxima seção fornece um pequeno tutorial para início rápido que você pode tomar como base.

2.1.1 Callgrind

O Callgrind é uma parte do Valgrind. Observe que ele era anteriormente chamado de Calltree, mas o nome foi abandonado.

O uso mais comum é anteceder a linha de comando para iniciar o seu aplicativo com o **`valgrind --tool=callgrind`**, como por exemplo

```
valgrind --tool=callgrind programa argumentos
```

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

Um uso mais avançado é gerar dados de auditoria sempre que uma determinada função de seu aplicativo é chamada. Por exemplo, para o Konqueror, para ver dados de auditoria somente para renderização de uma página web, você pode decidir gerar os dados sempre que você selecionar o item de menu **Ver** → **Recarregar**. Isto corresponde a uma chamada ao `KonqMainWindow::slotReload`. Use

```
valgrind --tool=callgrind --dump-before=KonqMainWindow::slotReload konqueror
```

Isto produzirá múltiplos arquivos de dados de auditoria com uma sequência numérica crescente no final dos nomes de arquivo. Um arquivo sem um número no final (terminando somente no PID do processo) também será produzido. Carregando este arquivo no KCachegrind, todos os outros serão também carregados, e podem ser vistos na **Visão Geral de Partes** e lista de **Partes**.

2.1.2 OProfile

O OProfile está disponível em sua página na internet. 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 usuário 'root', uma vez que todas as ações do sistema poderão ser observadas. Assim, as seguintes ações terão que ser feitas como 'root'. Primeiro, configure o processo de análise, usando a GUI **oprof_start** ou a ferramenta da linha de comando **opcontrol**. A configuração normal seria o modo de temporização (TBS, ver a introdução). Para iniciar a medida, execute o **`opcontrol -s`**. Depois execute o aplicativo em que está interessado e, a seguir, invoque um **`opcontrol -d`**. Isto irá apresentar os resultados das medidas nos arquivos 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 arquivos, um por cada programa que estava rodando no sistema. Cada um deles poderá ser carregado no KCachegrind por si só.

2.2 Fundamentos da Interface com o Usuário

Ao iniciar o KCachegrind com um arquivo de dados de auditoria como argumento, ou após carregar um através do **Arquivo** → **Abrir**, você verá um painel de navegação contendo a lista de funções à esquerda, e na parte principal à direita, uma área com visualizações para a função selecionada. Esta área de visualização pode ser configurada arbitrariamente para mostrar múltiplas visualizações de uma vez.

Na primeira vez, esta área estará dividida numa parte superior e outra inferior, tendo cada uma delas diferentes áreas que podem ser selecionadas 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 ativo: para o Callgrind, este é por exemplo 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 visão geral das características de execução deverá ser apresentada quando você selecionar a função `main` na lista da esquerda, e ver a visualização do gráfico de chamadas. Aí, você poderá ver as chamadas em curso no seu programa. Lembre-se que o gráfico de chamadas só mostra as funções com uma grande quantidade de eventos. Se fizer duplo-clique numa função do gráfico, ela irá mudar para mostrar as funções chamadas pela selecionada.

Para explorar mais a GUI, além deste manual, dê uma olhada na seção de documentação na página Web do projeto. Além disso, cada elemento gráfico do KCachegrind tem ajuda 'O Que é Isto?'.

Capítulo 3

Conceitos Básicos

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

3.1 O Modelo de Dados para Dados de Auditoria

3.1.1 Entidades de Custo

Contagem de custos de eventos (como Perdas na L2) são atribuídas a entidades de custo, que são itens com relacionamentos com o código-fonte ou estrutura de dados de um programa fornecido. Entidades de custo não somente podem ser código simples ou posições de dados, mas também posições de referência. Por exemplo, uma chamada tem uma fonte e um alvo, ou um endereço de dados pode ter um tipo de dado e uma posição do código onde 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 determinada linha de código, identificada pelo nome do arquivo de código e pelo número de linha, e que são executadas sob o contexto de uma determinada 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 arquivo `???`.

Função

Todas as linhas de código de uma determinada função compõem a função em si. Uma função é identificada pelo seu nome e pela sua localização no arquivo-objeto binário, se estiver disponível. A última é necessária porque os objetos binários de um único programa poderão conter funções com o mesmo nome (elas poderão ser acessadas, por exemplo, com o `dlopen'` ou `dlsym`; o editor de ligações durante a execução resolve as funções numa determinada ordem de objetos binários). Se uma ferramenta de análise não conseguir detectar o nome do símbolo de uma função, por exemplo porque a informação de depuração não está disponível, tanto é usado o endereço da primeira instrução executada, ou então o `???`.

Objeto Binário

Todas as funções cujo código esteja dentro do intervalo de um determinado objeto binário, seja ele o executável principal ou uma biblioteca dinâmica.

Arquivo de Código

Todas as funções cuja primeira instrução esteja mapeada numa linha do arquivo de código indicado.

Classe

Os nomes dos símbolos das funções estão tipicamente ordenados de forma hierárquica em espaços de nomes, por exemplo os 'namespaces' de C++, ou as classes das linguagens orientadas por objetos. Como tal, uma classe poderá conter funções da classe ou outras classes embebidas nela.

Parte de Análise

Alguma seção no tempo de uma execução da análise, com um dado ID de tarefa, ID de processo e uma linha de comando executada.

Como pode ser visto na lista, um conjunto de entidades de custo define normalmente outra entidade de custo, existindo uma hierarquia de inclusão das entidade de custo.

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, uma vez que isto não faz sentido num gráfico de chamadas. Assim, as sequências como o tratamento de exceções e os 'long jumps' do C terão que ser traduzidos em saltos na pilha de chamadas, de acordo com as necessidades.

3.1.2 Tipos de Eventos

Tipos de eventos arbitrários podem ser especificados nos dados de auditoria fornecendo-lhes um nome. Seu custo relacionado a uma entidade de custo é um inteiro de 64 bits.

Tipos de eventos os quais os custos são especificados em um arquivo de dados de auditoria são chamados eventos reais. Além disso, tipos podem especificar fórmulas para tipos de eventos calculados a partir de eventos reais, sendo chamados de eventos herdados.

3.2 Visualização de Estado

O estado da visualização de uma janela do KCachegrind inclui:

- o tipo primário e secundário dos eventos selecionados para mostrar,
- o agrupamento de funções (usado na lista da **Análise da Função** e na cor da entidade),
- as partes da análise cujos custos serão incluídos na visualização,
- uma entidade de custo ativa (por exemplo uma função selecionada a partir da barra de análise da função),
- uma entidade de custo selecionada.

Este estado influencia as visualizações.

As visualizações são sempre apresentadas apenas para a entidade de custo atualmente ativa. Quando uma determinada visualização não é apropriada para uma entidade de custo, fica desativada: por exemplo, ao selecionar um objeto ELF na lista de grupos a visão de anotação de código não faz sentido.

Por exemplo para uma função ativa, a lista de chamadas mostra todas as funções chamadas a partir da ativa. Alguém pode selecionar uma destas funções sem torná-la ativa. Se o gráfico de chamada é mostrado logo ao lado, ele automaticamente selecionará a mesma função.

3.3 Partes da GUI

3.3.1 Abas Laterais

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

- A **Análise de Função** é uma lista de funções mostrando custo inclusivo e exclusivo, contagem de chamadas, nome e posição de 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 padrão) 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 topo da página. Se existirem várias páginas, só uma estará ativa. O nome da entidade da página ativa é mostrado em negrito e determina a entidade de custo ativa da janela do KCachegrind.

3.3.3 Áreas de uma Página

Cada visão em aba pode conter quatro áreas de visão, nomeadas Superior, Direita, Esquerda e Inferior. Cada área pode conter múltiplas visualizações empilhadas. A parte visível de uma área é selecionada por uma barra de abas. Barras de abas da área superior e direita estão no topo, barras de abas da área inferior e esquerda estão na base. Você pode especificar que tipo de visualização deve ser colocado em qual área usando o menu de contexto das abas.

3.3.4 Visualização Sincronizada da Entidade Selecionada numa Página

Além de uma entidade ativa, cada aba tem uma entidade selecionada. Como a maioria dos tipos de visualização mostram múltiplas entidades com a ativa centrada, você muda o item selecionado navegando dentro de uma visualização (clicando com o mouse ou usando o teclado). Tipicamente, itens selecionados são mostrado em um estado destacado. Mudando a entidade selecionada em uma das visões em aba, todas as outras visualizações destacarão a nova entidade selecionada.

3.3.5 Sincronização entre Páginas

Se existirem múltiplas páginas, uma mudança de seleção em uma página faz um mudança de ativação na página seguinte, seja à direita ou acima dela.. Este tipo de ligação por exemplo deve permitir uma rápida navegação nos gráficos de chamadas.

3.3.6 Disposições

A disposição de todas as páginas de uma janela poderá ser salva (veja o item do menu **Ver** → **Disposição**). Após duplicar a disposição atual (**Ver** → **Disposição** → **Duplicar** (**Ctrl++**)) e alterar alguns tamanhos ou mudar uma visualização para outra área de uma página, você poderá mudar rapidamente entre a disposição antiga e a nova por meio da combinação **Ctrl**+← e **Ctrl**+→. O conjunto de disposições será salvo entre sessões do KCachegrind do mesmo comando analisado. Você poderá tornar o conjunto de disposições o padrão para as novas sessões do KCachegrind ou restaurar o conjunto de disposições padrão.

3.4 Abas Laterais

3.4.1 Auditoria Plana

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 dispendido, dependendo do tipo de grupo escolhido. A lista de grupos fica oculta quando o agrupamento está desligado.

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

3.4.2 Visão Geral das Partes

Na execução de uma análise, poderão ser produzidos vários arquivos de dados de análise que poderão ser carregados juntamente no KCachegrind. A barra de **Visão Geral das Partes** mostra estes arquivos, ordenados na horizontal de acordo com a hora de criação; os tamanhos dos retângulo são proporcionais ao custo dispendido nas partes. Você poderá selecionar uma ou várias partes para restringir os custos apresentados nas outras zonas do KCachegrind apenas para estas partes.

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

Modo de Partição

A repartição é exibida em grupos para uma parte de dados de análise, de acordo com o tipo de grupo selecionado. Por exemplo, se forem selecionados os grupos de objetos ELF, você irá ver retângulos coloridos para cada objeto ELF usado (biblioteca dinâmica ou executável), dimensionado de acordo com o custo nele dispendido.

Modo de Diagrama

Um retângulo mostrando o custo inclusivo da função ativa atual na parte é mostrado. Ele divide-se novamente para mostrar os custos inclusivos de suas chamadas.

3.4.3 Pilha de Chamadas

Isto é uma pilha de chamadas 'mais provável' puramente fictícia. Ela é construída iniciando com a função ativa atual e adicionar as chamadas e chamados com custo mais alto de cima para baixo.

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

3.5 Visões

3.5.1 Tipo de Evento

A lista **Tipo de Evento** mostra todos os tipos de custo disponíveis e o custo próprio correspondente e inclusivo da função ativa atual para o tipo de evento especificado.

Se escolher um tipo de evento na lista, você poderá alterar o tipo de custos apresentados em todo o KCachegrind para o tipo selecionado.

3.5.2 Lista de Chamadas

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

Visões de lista de chamada incluem:

- **Chamadores** Diretos
- **Chamados** Diretos
- **Todos os Chamadores**
- **Todas as chamadas**

3.5.3 Mapas

Uma visão de mapa em árvore do tipo de evento primário, acima e abaixo da hierarquia de chamadas. Cada retângulo colorido representa uma função; seu tamanho é aproximadamente proporcional ao custo gasto nela enquanto a função ativa estiver em execução (no entanto, existem restrições de desenho).

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

As opções de aparência poderão ser acessadas no menu de contexto. Para obter proporções de tamanho exatas, escolha a opção **Pular bordas incorretas**. Uma vez que este modo poderá tomar bastante tempo, o usuário poderá desejar limitar o nível máximo de encadeamento do desenho antes. O **Melhor** determina a direção 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 pelo teclado está disponível com as teclas de seta esquerda e direita para navegar por irmãos, e teclas de seta acima e abaixo para ir um nível aninhado acima e abaixo. **Enter** ativa o item atual.

3.5.4 Gráfico de Chamadas

Esta janela mostra o gráfico de chamadas em torno da função ativa. O custo apresentado é apenas o custo dispendido enquanto a função estava de fato rodando; isto é, o custo mostrado para o `main()` (se for visível) deverá ser o mesmo que o custo da função ativa, uma vez que faz parte do custo inclusivo do `main()` dispendido enquanto a função ativa estava em execução.

Para os ciclos, as setas de chamadas em azul indicam que esta é uma chamada artificial adicionada para desenhar corretamente o que, de fato, nunca ocorreu.

Se o grafo for maior que a área de desenho, é mostrada uma visão geral num dos lados. Existem opções de visualização semelhantes às do mapa de chamadas; a função selecionada está realçada.

3.5.5 Anotações

A lista de código ou assembler anotado mostra as instruções da linha de código ou desassembladas da função ativa atual junto com o custo (próprio) gasto ao executar o código de uma linha fonte ou instrução. Se existir uma chamada, linhas com detalhes sobre a chamada serão inseridas na fonte: o custo (inclusivo) gasto dentro de uma chamada, o número de chamadas que ocorreram, e o destino da chamada.

Selecione uma linha de informação de chamada para ativar o destino da chamada.

Capítulo 4

Referência de comandos

4.1 A janela principal do KCachegrind

4.1.1 Menu Arquivo

Arquivo → Novo (Ctrl-N)

Abre uma janela de nível superior vazia de onde você pode carregar dados de perfil. Esta ação não é realmente necessária, uma vez que **Arquivo → Abrir** lhe fornecerá uma nova janela de nível superior quando a atual já estiver exibindo algum dado.

Arquivo → Abrir (Ctrl-O)

Exibe o seletor de arquivos do KDE para escolher um arquivo de dados de auditoria a ser carregado. Se já existir algum dado sendo mostrado na janela de nível superior atual, ele abrirá uma nova janela. Se você deseja abrir dados de auditoria adicionais na janela atual, use **Arquivo → Adicionar**.

O nome dos arquivos 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 arquivos de dados de análise que pertençam uma execução de uma aplicação. Ao ler um arquivo que termine apenas em `pid`, os arquivos de dados eventualmente existentes para esta execução, mas sem terminações adicionais, são também carregados.

Se existir arquivos de dados de auditoria `cachegrind.out.123` e `cachegrind.out.123.1`, carregando o primeiro, o segundo será automaticamente carregado também.

Arquivo → Adicionar

Adiciona uma arquivo de dados de auditoria à janela atual. Com isto, você pode forçar que arquivos de dados múltiplos sejam carregados em uma mesma janela de nível superior mesmo se eles não vierem da mesma execução fornecida por uma convenção de nomeação de arquivos de dados de auditoria. Por exemplo, use-o para comparações lado a lado.

Arquivo → Recarregar (F5)

Recarrega os dados de auditoria. Isto é útil quando outro arquivo de dados de auditoria foi gerado pela execução de um aplicativo já carregado.

Arquivo → Sair (Ctrl-Q)

Sai do KCachegrind

Capítulo 5

Perguntas e respostas

Este documento pode ter sido atualizado depois da sua instalação. Você pode encontrar a última versão em http://docs.kde.org/ .

1. *Para que serve o KCachegrind? Eu não faço a mínima ideia.*

   O KCachegrind é útil no último estágio de desenvolvimento de software, chamado auditoria. Se você não desenvolve aplicativos, você não precisa do KCachegrind.

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

   Estes são atributos de custo para funções relativos a algum tipo de evento. Como funções podem chamar outras, faz sentido distinguir o custo da função propriamente dita ('Custo Próprio') e o custo incluindo todas as chamadas de funções ('Custo Inclusivo'). 'Próprio' é algumas vezes também referenciado como custo 'Exclusivo'.

   Assim, por exemplo para o `main()`, você sempre terá um custo inclusivo de cerca de 100%, visto que o custo próprio e negligenciado quando o trabalho real é feito em outra função.

3. *A barra de ferramentas e menu do meu KCachegrind está tão esquisita. Isto é normal?*

   O KCachegrind está provavelmente mal instalado no seu sistema. Recomenda-se que o compile com o prefixo de instalação igual à sua pasta de base do sistema KDE, como por exemplo o comando **`configure --prefix= /opt/kde4; make install`**. Se escolher outra pasta, como a `$HOME /kde`, você deverá apontar a variável de ambiente `KDEDIR` para esta pasta antes de executar o KCachegrind.

4. *Se eu der um duplo-clique em uma função abaixo da visão do **Gráfico de Chamadas**, ele mostra para a função `main()` o mesmo custo da função selecionada. Isto não é supostamente para ser 100% constante?*

   Você ativou 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 dispendida enquanto a função *ativa* está em execução, isto é, o custo mostrado para qualquer função nunca pode ser maior que o custo da função ativada.

Capítulo 6

Glossário

Entidade de Custo

Um item abstrato 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 (por exemplo, linha de código, função), a localização dos dados (por exemplo tipo dos dados acessados, o objeto de dados), a localização da execução (por exemplo, a tarefa ou processo) e os tuplos das posições acima indicadas (por exemplo, as chamadas, o acesso aos objetos pela instrução, os dados obtidos a partir da 'cache').

Custos do Evento

Soma de eventos de algum tipo de evento ocorrido durante a execução é relacionada a alguma entidade de custo. O custo é atribuído à entidade.

Tipo de Evento

O tipo de evento do qual custos podem ser atribuídos para uma entidade de custo. Aqui existem tipos de eventos reais e tipos de eventos herdados.

Tipo de Evento Herdado

Um tipo de evento virtual somente visível na visão, definido por uma fórmula a ser calculada a partir de tipos de eventos reais.

Arquivo de Dados de Análise

Um arquivo contendo dados medidos em um experimento de auditoria, ou parte de um, ou produzidos por processamento posterior de um rastreio. Seu tamanho é tipicamente linear em função do tamanho do código do programa.

Componente de Dados de Análise

Dados de um arquivo de dados de análise.

Experiência de Análise

Um programa executado sob supervisão de uma ferramenta de auditoria, produzindo possivelmente múltiplos arquivos de dados de auditoria a partir de partes ou linhas de execução.

Projeto de Análise

Uma configuração para experimentos de auditoria usados para um programa que será auditado, talvez em versões múltiplas. Comparações de dados auditados tipicamente só fazem sentido entre dados auditados produzidos em experimentos de um projeto de auditoria.

Auditoria

O processo de coletar informações estatísticas sobre as características de tempo de execução de um programa.

Tipo de Evento Real

Um tipo de evento que pode ser medido por uma ferramenta. Ele requer a existência de um sensor para o tipo de evento fornecido.

Rastreio

Uma sequência de eventos com estampas de tempo que ocorreram durante o rastreamento da execução de um programa. Seu tamanho é tipicamente linear em função do tempo de execução de um programa.

Componente de Rastreio

Ver "Componente de Dados de Análise".

Rastreamento

O processo de supervisionar a execução de um programa e armazenar eventos ocorridos ordenados por uma estampa de tempo em um arquivo de saída, o rastreio.

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.

Agradecimentos a todos os diferentes usuários que reportaram erros e sugestões.

Tradução de Marcus Gama marcus.gama@gmail.com

Apêndice A

Instalação

A.1 Como obter o KCachegrind

O KCachegrind faz parte do pacote kdesdk do KDE. Para as versões intermediárias menos suportadas, o Callgrind e a documentação futura, veja na página pessoal do projeto para obter mais instruções de instalação e compilação.

A.2 Requisitos

Para poder usar com sucesso o KCachegrind, você precisa do KDE 4.x. Para gerar os registros das análises, o Cachegrind ou o Calltree/Callgrind é recomendado.

A.3 Compilação e instalação

Para informações detalhadas de como compilar e instalar os aplicativos do KDE, visite a página KDE Techbase

Uma vez que o KDE usa o **cmake**, você não deve ter dificuldade em compilá-lo. Caso tenha algum problema, por favor, relate-o nas listas de discussão do KDE.

A.4 Configuração

Todas as opções de configuração encontram-se na janela de configuração ou nos menus de contexto dos gráficos.