Het handboek van KCachegrind

Oorspronkelijke auteur van de documentatie:
Josef Weidendorfer
Bijwerken en corrigeren: Federico Zenith
Vertaler/Nalezer: Freek de Kruijf
Vertaler: Ronald Stroethoff

Inhoudsopgave

1 Inleiding **6**

1.1 Profileren . . . 6
1.2 Profileringmethodes . . . 6
1.3 Profileringshulpmiddelen . . . 7
1.4 Visualisatie . . . 8

2 KCachegrind gebruiken **9**

2.1 Data-productie voor visualisatie . . . 9
2.1.1 Callgrind . . . 9
2.1.2 OProfile . . . 10
2.2 Gebruikersinterface . . . 10

3 Basis idee **11**

3.1 Het Data Model voor Profile Data . . . 11
3.1.1 Kostenelementen . . . 11
3.1.2 Gebeurtenistypen . . . 12
3.2 Toestand visualisatie . . . 12
3.3 Onderdelen van GUI . . . 13
3.3.1 Zijdocken . . . 13
3.3.2 Weergavezone . . . 13
3.3.3 Zones van een Tab . . . 13
3.3.4 Gesynchroniseerde weergave met geselecteerde element in een Tab . . . 13
3.3.5 Synchronisatie tussen Tabs . . . 13
3.3.6 Indelingen . . . 14
3.4 Zijdocken . . . 14
3.4.1 Kostenprofiel . . . 14
3.4.2 Overzicht van de onderdelen . . . 14
3.4.3 Aanroepstapel . . . 14
3.5 Weergaven . . . 15
3.5.1 Gebeurtenistype . . . 15
3.5.2 Aanroep lijsten . . . 15
3.5.3 Kaarten . . . 15
3.5.4 Graaf aanroepen . . . 15
3.5.5 Annotaties . . . 16

4 Overzicht van de opdrachten **17**
4.1 Het hoofdvenster van KCachegrind 17
4.1.1 Het menu Bestand 17

5 Vragen en antwoorden **18**

6 Woordenlijst **19**

7 Dankbetuiging en licentie **21**

Samenvatting

KCachegrind is een visualisatie hulpmiddel, geschreven met gebruik van KDE Frameworks.

Hoofdstuk 1

Inleiding

KCachegrind is een browser voor gegevens geproduceerd door profileringshulpmiddelen. Dit hoofdstuk legt uit waar profilering voor is, hoe het wordt gedaan en geeft enige voorbeelden van beschikbare profileringshulpmiddelen.

1.1 Profileren

Bij het ontwikkelen van een programma is vaak een van de laatste stappen het optimaliseren van de prestaties. Omdat het niet zinvol is zelden gebruikte functies te optimaliseren, omdat dat verspilling van tijd is, is het nodig om te weten in welk deel van het programma de meeste tijd wordt gebruikt.

Voor sequentiële code, volstaat het meestal om statistische data van een werkend programma zoals de gebruikte tijd in functies en coderegels te verzamelen. Dit heet profileren. Het programma werkt onder controle van een profilering-programma, wat aan het eind een overzicht geeft van de werking. Daarentegen worden voor parallelle code performance problemen meestal veroorzaakt doordat een processor aan het wachten is op data van een andere processor. Omdat de oorzaak van dit soort wachttijd niet makkelijk te achterhalen is, is het hier beter om ″timestamped event traces″ aan te maken. KCachegrind kan dit soort data niet visualiseren.

Na het analyseren van de geproduceerde profilering-data, is het makkelijk om de hotspots en knelpunten van de code te vinden: bijvoorbeeld, aannames over call counts zijn controleerbaar en identificeerbare code regions kunt u optimaliseren. Daarna, moet u het resultaat van de optimalisatie controleren met nog een profile run.

1.2 Profileringmethodes

Om precies de gepasseerde tijd te meten of om de gebeurtenissen tijdens de werking van een stuk code te noteren (bijv. een functie), moet er extra meet-code voor en achter het te onderzoeken code worden toegevoegd. Deze code leest de tijd, of een algemene gebeurtenis-teller, en berekent de verschillen. De originele code moet u dus voor de test wijzigen. Dit heet instrumentatie. Instrumentatie kan door de programmeur zelf, de compiler, of door het runtime systeem worden gedaan. Omdat interessante stukken code meestal genest zijn, beïnvloedt de overhead van de meting altijd de meting zelf. Instrumentatie moet u dus selectief toepassen en de resultaten moet u voorzichtig interpreteren. Dit maakt natuurlijk performance analyse door exacte metingen een complex proces.

Precieze meting is mogelijk met behulp van de hardware counters (met tellers die bij iedere tik omhoog gaan) die in moderne processors aanwezig zijn, die bij iedere gebeurtenis een stap worden verhoogd. Omdat we gebeurtenissen aan stukken code willen toevoegen, zouden we (zonder de tellers) zelf iedere gebeurtenis moeten bijhouden door voor het geselecteerde stuk code een teller te verhogen. Dit in software uitvoeren is, natuurlijk, niet mogelijk; maar, aangenomen dat de gebeurtenissen netjes over de code verdeelt is, kunnen we alleen maar elke n-th gebeurtenis te kijken en niet naar elke gebeurtenis, een meetmethode is ontwikkelt waarvan de overhead inschakelbaar is: dit heet Sampling. Op tijd gebaseerde Sampling (TBS) gebruikt een timer om regelmatig naar een programma-teller voor het creëren van een histogram voor de programma code. Op gebeurtenissen gebaseerde Sampling (EBS) gebruikt de hardware tellers van moderne processors, en gebruikt een modus waar een interrupt handler wordt aangeroepen als de teller op nul is voor het genereren van een histogram van de bijbehorende gebeurtenis distributie: in de handler, zal de gebeurtenis teller altijd weer worden geïnitialiseerd tot het n van de meetmethode. Het voordeel van sampling is dat de code niet gewijzigd hoeft te worden, maar het blijft een compromis: de bovengenoemde aanname zal meer correct zijn als n klein is, maar hoe kleiner n is , hoe groter de overhead van de interrupt handler.

Een andere meetmethode is het simuleren van gebeurtenissen in het computer systeem bij het uitvoeren van een gegeven code, bijv. executie gedreven simulatie. De simulatie is altijd afgeleid van een min of meer accuraat machine model; maar, met zeer gedetailleerde machine modellen, geeft dat vrij nauwkeurige benaderingen van de werkelijkheid, de simulatie tijd kan in de praktijk onacceptabel groot zijn. Het voordeel van simulatie is dat willekeurig ingewikkelde meting/simulatie code aan een gegeven code zonder storende resultaten kan worden toegevoegd. Door dit net voor de uitvoer te doen (runtime instrumentatie genaamd), met gebruik van de originele binary, is dit erg handig voor de gebruiker: een her-compilatie is dan niet nodig. Simulatie is bruikbaar als u alleen gedeeltes van machine met een eenvoudig model uitvoert; een ander voordeel is dat de door eenvoudige modellen geproduceerde resultaten vaak makkelijker te begrijpen zijn: vaak is het probleem met echte hardware dat in de resultaten elkaar overlappende effecten uit verschillende gedeeltes van de machine zitten.

1.3 Profileringshulpmiddelen

Het meest bekend is het GCC profiling programma gprof: u moet het programma compileren met de optie `-pg`; het laten werken van het programma genereert een bestand `gmon.out`, dat u vervolgens in een door mensen leesbaar formaat kunt omzetten met **gprof**. Een nadeel is de benodigde her-compilatie om de executable voor te bereiden, die statistisch gelinkt moet zijn. De hier gebruikte methode is compiler-genereerde instrumentatie, die alle tussen functies uitgevoerde call arcs en bijbehorende call tellers meet, in samenwerking met TBS, die een histogram van de tijd-verdeling over de code geeft. Door beide stukken informatie te gebruiken, is het mogelijk om heuristisch alle door een functie gebruikte tijden te berekenen, bijv. de tijd die een functie inclusief alle aangeroepen functies heeft gebruikt

Voor nauwkeurige metingen van gebeurtenissen, bestaan er bibliotheken met functies die hardware performance tellers kunnen uitlezen. Het meest bekend hier is de PerfCtr patch voor Linux®, en de architectuur onafhankelijke bibliotheken PAPI en PCL. Maar toch, nauwkeurige metingen heeft instrumentatie van code nodig, zoals hierboven al gezegd. Naar keuze gebruikt u de bibliotheken zelf of gebruikt u automatische instrumentatie systemen zoals ADAPTOR (voor FORTRAN broncode instrumentatie) of DynaProf (code injectie via DynInst).

OProfile is een systeembreed profileringshulpmiddel voor Linux® met gebruik van Sampling.

In veel aspecten is het gebruik van Cachegrind of Callgrind een handige manier van Profiling, deze zijn simulators die de runtime instrumentatie framework Valgrind gebruiken. Omdat u hier geen toegang tot de hardware tellers hoeft te hebben (vaak onhandig met de huidige Linux® installaties), en de binaries die u wilt profileren ongewijzigd kunnen blijven, is het een goed alternatief voor andere profileringshulpmiddelen. Het nadeel van simulatie - vertraging - kunt u verminderen door de simulatie alleen uit te voeren op de interessante programma-gedeeltes, en misschien alleen een paar iteraties van een loop. Zonder meting/simulatie instrumentatie, geeft

het gebruik van uitsluitend Valgrindalleen een vertraging 's factor van 3 tot 5. Wanneer alleen de call grafieken en call tellers interessant zijn , dan kunt u de cache simulator uitschakelen.

Cache simulatie is de eerste stap in het benaderen van reële tijden, omdat runtime erg gevoelig is voor het gebruik van de zogenoemde *caches*, kleine en snelle buffers in moderne systemen die herhaalde toegang tot zelfde stukken hoofd geheugen versnellen. Cachegrind voert cache simulatie uit door het catchen van geheugen-toegangen. In de geproduceerde data is ook het aantal uitlezingen van instructie/data geheugen mory accesses en gemiste eerste- en tweede-level cache uitlezingen, en de relatie tot de regels en functies in de broncode van het programma. Door het combineren van de aantal gemiste en gebruik van de miss latencies van standaard processors, is het mogelijk om een schatting van de gebruikte tijd te geven.

Callgrind is een uitbreiding van Cachegrind dat grafiek van aanroepingen onmiddellijk opbouwt, bijv. hoe deze functies elkaar aanroepen en hoeveel gebeurtenissen tijdens een aangeroepen functie gebeuren. Maar de profile data kan ook voor elke thread call chain context apart worden verzameld. Het kan profiling data op instructie niveau geven voor annotatie van gedisassembleerde code.

1.4 Visualisatie

Profileringshulpmiddelen geven standaard grote hoeveelheden data. De wens om makkelijk omhoog en omlaag te bladeren door de call grafiek, samen met snel omschakelen van de functie-overzicht en de weergaven van verschillende gebeurtenistypen, doen verlangen naar een GUI programma die deze taak kan uitvoeren.

KCachegrind is een visualisatiemiddel voor profileringsdata dat deze wensen kan vervullen. Ondanks dat het geprogrammeerd is met doorzoeken van de data van Cachegrind en Calltree in gedachte, zijn er converters beschikbaar om data weer te geven dat afkomstig is van andere hulpmiddelen. In de appendix is een beschrijving van het bestandsformaat van Cachegrind/-Callgrind te vinden.

Naast een lijst met functies die gesorteerd zijn op kosten, en optioneel gegroepeerd op bronbestand, shared library of C++ class, heeft KCachegrind ook verschillende weergaven-mogelijkheden voor een geselecteerde functie, namelijk:

- Weergave van een call-grafiek, die een sectie van de call-grafieken rond de geselecteerde functie toont.
- een boom-weergave, waarmee de relaties van geneste aanroepen worden weergegeven, samen met de bijbehorende kosten voor snelle visuele detectie van problematische functies.
- Weergave van annotatie van broncode en disassembler, waarmee u details van de kosten ten opzichte van regels en assembler- instructies kunt bestuderen.

Hoofdstuk 2

KCachegrind gebruiken

2.1 Data-productie voor visualisatie

Eerst moet u met behulp van een profilering-hulpmiddel performance data genereren door aspecten van het werkende programma te meten. Cachegrind heeft zelf geen profileringshulpmiddel, maar u kunt het goed samen met Callgrind gebruiken, en door een converter te gebruiken, kunt u het ook gebruiken om de data te visualiseren dat door OProfile is geproduceerd. Alhoewel het profileren met deze hulpmiddelen buiten de scoop van deze handleiding valt, geeft de volgende sectie een korte uitleg over daarmee te beginnen.

2.1.1 Callgrind

Callgrind is een onderdeel van Valgrind. Merk op dat het in het verleden Calltree werd genoemd , maar deze naam was misleidend.

De meest voorkomende gebruik is het toevoegen van een voorvoegsel aan de commandoregel bij het starten van uw programma met **`valgrind --tool=callgrind`** , zoals in:

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

Bij het beëindigen van het programma, zal er een bestand genaamd `callgrind.out.pid` worden gegenereerd, dat u in KCachegrind kunt laden.

Meer geavanceerd gebruik is het dumpen van profielgegevens bij het aanroepen van een opgeven functie in uw programma. B.v. voor Konqueror, om alleen profielgegevens te zien van het renderen van een Web pagina, kunt u de beslissing nemen om de data te dumpen als u het menu-item **Beeld** → **Herladen** selecteert. Dit is overeenkomstig met een aanroep van `KonqMainWindow::slotReload`. Gebruik:

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

Dit zal meerdere profielgegevens-bestanden (met een extra volgnummer aan het eind van de bestandsnaam) produceren. Een bestand zonder een dergelijk nummer aan het eind (alleen eindigend op het proces-PID) wordt ook geproduceerd; door dit bestand in KCachegrind te laden, zullen alle andere ook worden geladen en zijn vervolgens zichtbaar in **Profieloverzicht** en de**Profiel**lijst.

2.1.2 OProfile

OProfile is beschikbaar via zijn home pagina. Volg de installatie instructies op de Web site, maar, voordat u dat doet, controleer eerst of uw distributie het niet al als pakket levert (zoals SuSE®).

Het systeemmbreed profileren is alleen voor de gebruiker root toegestaan, omdat u dan alle acties in het systeem kunt observeren; daarom moet u het volgende als root uitvoeren. Configureer eerst het profilering-proces met behulp van de GUI **oprof_start** of het programma voor de commandoregel **opcontrol**. Standaard configuratie moet de timer mode zijn (TBS, zie introductie). Om de meting te starten, start **`opcontrol -s`**. Start vervolgens het programma waarin u bent geïnteresseerd en, geef aan het eind het commando **`opcontrol -d`**. Dit zal de meetresultaten wegschrijven naar bestanden in de map `/var/lib/oprofile/samples/` . Om de data te kunnen visualiseren in KCachegrind, geef in een lege map het volgende commando:

```
opreport -gdf | op2callgrind
```

Dit zal een heleboel bestanden produceren, een voor elke programma dat in het systeem draaide. Elk kunt u apart in KCachegrind laden.

2.2 Gebruikersinterface

Als u KCachegrind start met een profielgegevensbestand als argument, of na het via **Bestand** → **Openen** van laden van een ervan, krijgt u een navigatie-paneel te zien met de functie-lijst aan de linkerkant; en, aan de rechterkant het hoofdvenster, een ruimte met vensters voor een geselecteerde functie. Deze vensterruimte kunt u willekeurig indelen om meerdere vensters tegelijk te kunnen zien.

Bij de eerste keer starten zal deze ruimte verdeelt zijn in een boven en een onder gedeelte, elk met verschillende tab-selecteerbare vensters. Om de vensters te verplaatsen, gebruikt u het tabs context menu, en pas de splitters tussen de vensters aan. Om snel tussen de verschillende beeld layouts te schakelen, gebruikt u **Beeld** → **Layout** → **Ga naar volgende** (**Ctrl+**→) en **Beeld** → **Layout** → **Ga naar vorige** (**Ctrl+**←).

Het active gebeurtenistype is belangrijk voor visualisatie: voor Callgrind, is dit, bijvoorbeeld, gemiste cache of cyclus schatting; for OProfile, dit is voor de eenvoudigste gevallen 'Timer'. U kunt het event type via een keuzelijst in de werkbalk of in het **Gebeurtenistype** venster. Een eerste overzicht van de runtime karakteristieken krijgt u te zien als u de functie `main` in de linker lijst selecteert; kijk vervolgens naar het venster met de call-grafiek. Daar ziet u de calls die in uw programma voorkomen. Merk op dat het call graph venster alleen functies toont die een groot aantal keren voorkomen. Door op een functie in de grafiek te dubbelklikken, krijgt u de door de geselecteerde functie aangeroepen functies te zien krijgen.

Om de GUI verder te verkennen, kunt u ook behalve deze handleiding ook in de sectie documentatie van de Web site kijken. Elke widget in KCachegrind heeft ook een 'Wat is dit' hulp.

Hoofdstuk 3

Basis idee

Dit hoofdstuk legt enkele basisideeën van KCachegrind uit, en introduceert in het interface gebruikte termen.

3.1 Het Data Model voor Profile Data

3.1.1 Kostenelementen

De kosten van gebeurtenistypes (zoals L2 Misses) worden ingedeeld als kostenelementen, wat items zijn met een relatie tot broncode of data structuren van een gegeven programma. Kostenelementen kunnen niet alleen eenvoudige code of data posities zijn, maar ook positie tuples. Bijvoorbeeld, een call heeft een bron en een doel, of een data adres kan een data type en een code positie waar zijn allocatie gebeurt.

De kostenelementen die bij KCachegrind bekend zijn, volgen hieronder. Simple Positions:

Instruction
: Een assembler instructie op een opgegeven adres.

Source Line of a Function
: Alle instructies die de compiler (via debug informatie) mapt naar een opgegeven regel in de broncode (gespecificeerd door broncodebestand en regelnummer), en die in de context van een functie worden uitgevoerd. Dit laatste is nodig omdat regelcode in een inlined functie in de context van meerdere functies kan verschijnen. Instructie zonder enige mapping naar een daadwerkelijke coderegel worden gemapt naar regelnummer 0 in bestand `???`.

Function
: Alle coderegels van een opgegeven functie horen bij de functie zelf. Een functie is gespecificeerd door zijn naam en de locatie in een binair object (indien beschikbaar). Dit laatste is nodig omdat elk binair object van een enkel programma functies met dezelfde naam kan hebben (deze zijn toegankelijk met bijv. met `dlopen` of `dlsym`; de runtime linker zoekt functies in een opgegeven zoek-volgorde voor gebruikte binary objects op). Als een profileringshulpmiddel het symboollabel van een functie niet kan vinden, bijv. omdat debug informatie niet beschikbaar is, ofwel het adres van de eerst uitgevoerde instructie standaard is gebruikt, of `???`.

Binary Object
: Alle functies waarvan de code is binnen de range van een opgegeven binar object, of van het hoofd executable of een gedeelde library.

Source File
: Alle functies waarvan de eerste instructie is gemapped naar een regel van de opgegeven broncodebestand.

Class
: Symbol namen van functies zijn standaard hiërarchisch geordend in name spaces, bijv. C++ namespaces, of classes van object-georiënteerde talen; dus, een class kan functies van de class of ingebedde classes zelf hebben.

Profile Part
: Een tijd sectie van een profile run, met een opgegeven thread ID, process ID, en uitgevoerde command line.

Zoals u in de lijst kan zien, een set van kostenelementen veroorzaakt weer andere kostenelementen; dus, er is een inclusieve hiërarchie van kostenelementen.

Positieons tuples:

- Aanroep van instructie adres naar doel functie.
- Aanroep van doel functie vanuit coderegel.
- Aanroep van doel functie vanuit bron functie.
- (On)voorwaardelijke sprong van bron naar doel instructie.
- (On)voorwaardelijke sprong van bron naar doel regel.

Sprongen tussen functies zijn niet toegestaan, omdat dit geen zin heeft in een call graph; dus, constructs zoals exception handling en long jumps in C moeten zo nodig worden vertaald naar popping de call stack.

3.1.2 Gebeurtenistypen

U kunt willekeurige gebeurtenistypen opgeven in de profile data door ze een naam te geven. De kosten daarvan vergeleken met een kostenelement is een 64-bit integer.

Gebeurtenistypen waarvan de kosten zijn opgegeven in een profile data bestand worden echte events genoemd. Daarnaast kunt u formules specificeren voor uit echte gebeurtenissen berekende event-types, die inherited events worden genoemd.

3.2 Toestand visualisatie

De toestand visualisatie van een KCachegrind venster is inclusief:

- de voor weergave gekozen primaire en secundaire gebeurtenistype,
- de functie groepering (gebruikt in het **Functieprofiel** lijst en element kleuring),
- de profielgegevens waarvan de kosten ook in de visualisatie zichtbaar moeten zijn,
- een actieve kostenelement (bijv. een uit het functie profiel zijdock geselecteerde functie),
- een geselecteerde kostenelement.

Deze toestanden beinvloedt de weergaven.

Weergaven worden altijd voor een kostenelement getoond, de actieve. Als een opgegeven weergave ongeschikt is voor een kostenelement, dan is het uitgeschakeld: bij het selecteren van bijv. een ELF object in de groep-lijst, is een source annotatie niet zinvol.

Bijvoorbeeld, voor een een actieve functie, toont de lijst met aangeroepenen alle functies die door de actieve zijn aangeroepen: u kunt een van deze functies selecteren zonder hem actief te maken. Daarnaast, als de call graph daarnaast wordt getoond, dan zal deze automatisch dezelfde functie selecteren.

3.3 Onderdelen van GUI

3.3.1 Zijdocken

Zijdocken zijn zijvensters die u aan elke rand van een KCachegrind-venster kunt plaatsen. Zij hebben altijd een lijst met kostenelementen die op een bepaalde volgorde staan.

- De **Functieprofiel** is een lijst met functies die de inclusive en exclusive kosten, getelde aanroepen, naam en positie van functies toont.
- **Overzicht van de onderdelen**
- **Aanroepstapel**

3.3.2 Weergavezone

De weergavezone, standaard het rechter gedeelte van het hoofdvenster van KCachegrind, heeft een (standaard) of meerdere tabs, horizontaal of verticaal verdeelt. Elke tab heeft meerdere weergaven met elk een kostenelement tegelijk. De naam van dit element is aan de bovenkant van de tab te zien. Als er meerdere tabs zijn, dan is er maar een actief. In de actieve tab is de elementnaam vetgedrukt, en bepaalt de kostenelement van het KCachegrind -venster.

3.3.3 Zones van een Tab

Elke tab kan tot vier weergavezones hebben, genaamd Top, Right, Left, en Bottom. Elke zone kan meerdere opgestapelde weergaven hebben. Het zichtbare gedeelte van een zone is selecteerbaar door middel van een tabbalk. De tab balken van de rechter en bovenste gebieden zijn aan de bovenkant; de tab balken van de linker en onderste gebieden zijn aan de onderkant. U kunt instellen door middel van de contextmenu´s van de tab welk soort weergave in welke zone gaat.

3.3.4 Gesynchroniseerde weergave met geselecteerde element in een Tab

Naast een actief element, heeft elke tab een geselecteerde element. Omdat de meeste weergavetypes meerdere elementen tonen met de actieve op de een of andere manier gecentreerd, kunt u het geselecteerde item wijzigen door binnen de weergave te navigeren (door met de muis te klikken of door het toetsenbord te gebruiken). Standaard worden geselecteerde items als gemarkeerd getoond. Door de geselecteerde item in een van de weergaven in een tab te wijzigen, zijn in alle andere weergaven het nieuw geselecteerde item ook gemarkeerd.

3.3.5 Synchronisatie tussen Tabs

Als er meerdere tabs zijn, dan lijdt een selectiewijziging in een tab tot wijziging in activering in de volgende tab, onafhankelijk of dit rechts of onder daarvan is. Deze manier van linken zou, bijvoorbeeld, het mogelijk maken om snel te bladeren in call graphs.

3.3.6 Indelingen

De layout van alle tabs in een venster kunt u opslaan (**Beeld** → **Layout**). Na het dupliceren van de geselecteerde layout (**Beeld** → **Layout** → **Dupliceren** (**Ctrl++**)) en het wijzigen van enkele afmetingen of het verplaatsen van een weergave naar een plek in een tab, kunt u snel heen en weer schakelen tussen de oude en de nieuwe layout via **Ctrl**+← en **Ctrl**+→. De verzameling layouts zullen tussen de sessies van KCachegrind met hetzelfde profilering-commando worden opgeslagen. U kunt de huidige verzameling layouts de standaard maken voor nieuwe KCachegrind sessies, of de standaard layout verzameling weer terugzetten.

3.4 Zijdocken

3.4.1 Kostenprofiel

De **Kostenprofiel** heeft een lijst met groepen en een lijst met functies. In de lijst met groepen vindt u alle groepen waar kosten zijn gemaakt, afhankelijk van de gekozen type groep. De lijst met groepen is verborgen als groeperen is uitgeschakeld.

De functie-lijst toont de functies van de geselecteerde groep (of alle functies als groeperen is uitgeschakeld), gesorteerd volgens een bepaalde kolom, bijv. inclusief of zelf gespendeerde kosten. Er is een maximum aantal functies die te zien zijn in de lijst, instelbaar in **Instellingen** → **KCachegrind instellen**.

3.4.2 Overzicht van de onderdelen

In een profile run, is het mogelijk om meerdere profile data bestanden te produceren, die u samen in KCachegrind kunt laden. De zijdock **Overzicht van geladen trace-profielbronnen** toont deze, horizontaal geordend overeenkomstig het moment van creatie; het formaat rechthoek is overeenkomstig met de gespendeerde kosten van elk onderdeel. U kunt een of meerde profielen selecteren om de in de andere KCachegrind-weergaven zichtbare kosten te beperken.

De details zijn verder onderverdeeld in een detail- en een inclusive kosten split mode:

Modus voor partitionering

De details zijn in groepen voor een profielgegevens te zien, overeenkomstig de geselecteerde type groep. Bijvoorbeeld, als u ELF object groepen zijn selecteert, dan ziet u gekleurde rechthoeken voor elk gebruikt ELF object (shared library of executable), de grootte overeenkomstig de kosten daarvan.

Diagrammodus

Een rechthoek toont de inclusieve kosten van de geselecteerde actieve functie in de getoonde details. Ook dit is verdeelt in de inclusieve kosten van de aangeroepenen.

3.4.3 Aanroepstapel

Dit is een puur fictieve 'meest waarschijnlijke' call stack. Het wordt opgebouwd bij de start van de huidige actieve functie, en voegt de aanroepers en aangeroepenen toe met de hoogste kosten bovenaan.

De kolommen **Kosten** en **Aanroepen** tonen de gebruikte kosten voor alle aanroepen van de functie in de regel erboven.

3.5 Weergaven

3.5.1 Gebeurtenistype

De **Gebeurtenistype** lijst toont alle beschikbare kostentypes en de bijbehorende eigen en inclusieve kosten van de huidige actieve functie voor dat gebeurtenistype.

Door uit de lijst een gebeurtenistype te kiezen, kunt u in de gehele KCachegrind de getoonde kostentype wijzigen naar de geselecteerde.

3.5.2 Aanroep lijsten

Deze lijst toont de calls naar en van de huidige actieve functie. Met **Alle Callers** en **Alle Aangeroepenen** betekenen die functies die bereikbaar zin in de richting van het aanroepen, zelfs als er andere functie er tussen zitten.
De aanroeplijst is inclusief:

- Directe **Aanroepers**
- Directe **Aangeroepenen**
- **Alle aanroepers**
- **Alle aangeroepenen**

3.5.3 Kaarten

Een boomstructuur overzicht van het primaire gebeurtenistype, omhoog of omlaag in de call hierarchy. Elke gekleurde rechthoek stelt een functie voor; de grootte komt ongeveer overeen met de gespendeerde kosten als de functie actief is (maar, er zijn beperkingen in de weergave).

Voor de **Aanroeperskaart**, toont de graaf de geneste hierarchy van alle aanroepers in de huidige geselecteerde functie; voor de **Aangeroepenenkaart**, toont het voor alle aangeroepenen.

Weergave instellingen kunt u in het context menu vinden. Om precieze afmetingen te krijgen, selecteert u **Incorrecte Borders overslaan**. omdat deze modus veel computer-tijd kan gebruiken, wilt u wellicht het maximum aantal drawn nesting level beperken. **Best** determinates the split direction for children from the aspect ratio of the parent. **Altijd Best** beslist over de overblijvende ruimte voor elk broer. **Ignore Proportions** ruimte in beslag neemt voor de naam van de functie die van de kinderen afgaat. Merk op dat de proporties van de afmetingen erg kunnen afwijken.

Toetsenbord navigatie is beschikbaar met de links en rechts pijltjestoetsen om door de neefjes te wandelen en de omhoog en omlaag pijltjestoetsen om een genest niveau omhoog en omlaag te gaan. **Enter** selecteert het huidige item.

3.5.4 Graaf aanroepen

Deze weergave toont de aanroep graaf rond de actieve functie. De getoonde kosten zijn alleen de gebruikte kosten als de actieve functie daadwerkelijk heeft gewerkt, bijv. de getoonde kosten voor `main()` (als het zichtbaar is) zou hetzelfde moeten zijn als de kosten van de actieve functie, omdat dit het gedeelte van de gebruikte inclusieve kosten zijn van `main()` terwijl het een werkend actieve functie was.
Voor cycles, geven blauwe call pijlen aan dat dit een kunstmatige call is, die nooit echt is gebeurt, toegevoegd voor een correcte weergave.

Als de graaf groter is dan de weergavezone, dan krijgt u een bird's eye view aan de zijkant te zien is. De opties voor weergave zijn vergelijkbaar met die van de call maps; de geselecteerde functie is gemarkeerd.

3.5.5 Annotaties

De annotaties van broncode of assembler lijsten tonen de broncoderegels of disassembled instructies van de huidige actieve functie samen met (zelf) gebruikte kosten voor het uitvoeren van de code van een broncoderegel of instructie. Als er een call was , dan zijn de regels met details over de call aan de bron toegevoegd: de (inclusieve) in de call gebruikte kosten, het aantal uitgevoerde calls, en het doel van de call.

Selecteer dergelijke informatieregel over een call om het call-doel te activeren.

Hoofdstuk 4

Overzicht van de opdrachten

4.1 Het hoofdvenster van KCachegrind

4.1.1 Het menu Bestand

Bestand → Nieuw (Ctrl+N)

Opent een leeg top-level venster waarin u profielgegevens kunt laden. Deze actie is niet echt noodzakelijk, omdat **Bestand → Openen** u een nieuw top-level venster geeft indien de huidige al gegevens toont.

Bestand → Openen (Ctrl+O)

Opent het KDE-bestand venster voor het kiezen welk profielbronbestand u wilt laden. Als er al data zichtbaar is in huidige top-level venster, dan opent een nieuw venster; als u extra profielgegevens in in het huidige venster openen, dan moet u **Bestand → Toevoegen** gebruiken.

De namen van de profielbronbestanden eindigen meestal in `.pid.part-threadID`, waar *`part`* en *`threadID`* optioneel zijn. *`pid`* en *`part`* worden gebruikt in het geval dat meerdere profielbronbestanden bij een programma run horen. Door het laden van een bestand dat met alleen `pid` eindigt, zullen de ander bestanden die bij deze run horen met extra karakters eindigen ook geladen worden.

Als er de profielbronbestanden `cachegrind.out.123` en `cachegrind.out.123.1` bestaan , dan zal u door de eerste te laden, automatisch ook de tweede laden.

Bestand → Toevoegen

Voegt een profielbronbestand toe aan het huidige venster. Door dit te gebruiken, kunt u het laden van meerdere data-bestanden in hetzelfde top-level venster forceren zelfs als ze van dezelfde run zijn, zoals ingesteld door de profielbronbestand-conventie. U kunt dit bijvoorbeeld gebruiken voor het naast elkaar vergelijken.

Bestand → Herladen (F5)

Herlaad de profielgegevens. Dit is handig als nog een profielbronbestand is gegenereerd voor een al geladen programma run.

Bestand → Afsluiten (Ctrl+Q)

Beëindigt KCachegrind

Hoofdstuk 5

Vragen en antwoorden

1. *Waar is KCachegrind voor? Ik heb geen idee.*

   KCachegrind is behulpzaam in een late fase van software ontwikkeling, profilering genaamd. Als u geen programma´s ontwikkelt, dan heeft u KCachegrind niet nodig.

2. *Wat is het verschil tussen **Incl.** en **Self**?*

   Dit zijn kostenelementen voor functies wat betreft sommige gebeurtenistypen. Omdat functies elkaar kunnen aanroepen (call), is het zinnig om onderscheidt te maken tussen de kosten van de functie zelf ('Self Cost') en de kosten incusief alle aangeroepen functies ('Inclusieve kosten'). Aan 'Self' wordt soms ook gerefereerd als 'Exclusieve' kosten.

   Dus, bijvoorbeeld, voor `main()`, heeft u altijd inclusieve kosten van bijna 100%, waar de kosten zelf bijna verwaarloosbaar is omdat het echte werk in andere functies wordt gedaan.

3. *Als ik dubbelklik op een functie lager in de **Call Graph** weergave, dan toont voor de functie `main()` dezelfde kosten als de gekozen functie. Zou dit niet constant 100% moeten zijn?*

   U een heeft een functie onder `main()` geactiveerd, die uiteraard minder kost dan `main()` zelf. Voor elke functie, wordt alleen het gedeelte van de kosten getoond als de *geactiveerde* functie werkend is; dat houdt in, dat de getoonde kosten voor elke functie kan nooit hoger zijn dan de kosten van de geactiveerde functie.

Hoofdstuk 6

Woordenlijst

Kostenelementen
Een abstract item in relatie tot broncode waaraan gebeurteniskosten kunnen worden toegewezen. Dimensies voor kostenelementen zijn code locatie (bijv. coderegel, functie), data locatie (bijv. accessed data type, data object), executie locatie (bijv. thread, process), en tuples of triples van de eerder genoemde posities (bijv. calls, object access van statement, evicted data van cache).

Gebeurteniskosten
Totaal aantal gebeurtenissen van een bepaalde gebeurtenistype die optreden tijdens de uitvoering en die een relatie met een bepaalde kostenelement hebben. De kosten hebben een relatie met het element.

Gebeurtenistype
Het soort gebeurtenis waarbij kosten aan een kostenelement kunnen worden toegewezen. Er zijn echte gebeurtenistypen en inherited event types.

Inherited gebeurtenistype
Een virtueel gebeurtenistype alleen zichtbaar in de weergave, gedefinieerd door een formule zodat het uit echte gebeurtenistypen kan worden berekent.

Profielgegevensbestand
Een bestand met in een profilering-experiment gemeten data, of een gedeelte daarvan, of geproduceerd door post-processing van een trace. De grootte daarvan is standaard linear met de grootte van de code van het programma.

Profile Data Part
Data van een profielgegevensbestand

Profilering-experiment
Een programma dat werkt onder supervisie van profileringsprogramma, mogelijk meerdere profielgegevensbestanden van gedeeltes of threads van de run producerend.

Profilering project
Een configuratie voor profilering experimenten gebruikt om een programma te profileren, misschien in meerdere versies. Vergelijkingen van profilering data is standaard alleen zinvol tussen profilering data geproduceerd in experimenten van een profilering project.

Profileren
Het proces van het verzamelen van statistische informatie over runtime karakteristieken van werkende programma´s.

Echt gebeurtenistype

Een gebeurtenistype dat meetbaar met een hulpprogramma. Dit vereist een sensor voor het gegeven gebeurtenistype.

Trace

Een rij van timestamped gebeurtenissen die optraden tijdens het tracen van programma run. De grootte is standaard linear met de executie tijd van de programma run.

Trace Part

Zie "Profile Data Part".

Tracing

Het proces van supervising een programma run en het opslaan van de gebeurtenissen, gesorteerd op een timestamp, in een output-bestand, de trace.

Hoofdstuk 7

Dankbetuiging en licentie

Dank aan Julian Seward voor zijn uitstekende Valgrind, en Nicholas Nethercote voor de uitbreiding Cachegrind. Zonder deze programma´s zou KCachegrind niet bestaan. Sommige ideeën voor deze GUI zijn van hun afkomstig.

Dank voor alle bug-rapporten en suggesties van verschillende gebruikers.

Op- of aanmerkingen over de vertalingen van de toepassing en haar documentatie kunt u melden op http://www.kde.nl/bugs.

Dit document is vertaald in het Nederlands door Freek de Kruijf freekdekruijf@kde.nl.

Dit document is vertaald in het Nederlands door Ronald Stroethoff stroet43@zonnet.nl.