# Laufzeitgarantien auf Prozessoren mit Caches

Lehrstuhl Prof. Dr. Reinhard Wilhelm
Fachbereich Informatik / Universität des Saarlandes
Postfach 15 11 50 / D-66041 Saarbrücken
Fon: 0681 302 3434 Fax: 0681 302 3065
wilhelm@cs.uni-sb.de
http://www.cs.uni-sb.de/~wilhelm

## 1 Eingebettete Echtzeitsysteme

Echtzeitsysteme sind dadurch charakterisiert, daß sie ihre Funktionen innerhalb vorgegebener Zeitgrenzen erbringen müssen. Diese Zeitgrenzen werden meist durch die physikalische Umgebung diktiert, in welche das System eingebettet ist. Eine Verletzung der Zeitbedingungen verursacht eventuell großen Schaden an Mensch und Umgebung. Die Einhaltung der Zeitbedingungen durch das eingebettete Echtzeitsystem muß vom Entwickler garantiert werden. Er prüft, ob alle Prozesse, aus denen das System besteht, auf der ausgewählten Hardware unter Beachtung ihrer Zeitbedingungen ausgeführt werden können (Planbarkeitsanalyse). Hierzu muß die Ausführungszeit der Prozesse für den schlechtesten Fall bekannt, d. h. durch den Übersetzer berechnet sein. Das ist bei den einfachen Microcontrollern, die häufig in Echtzeitsystemen eingesetzt werden, kein Problem.

Moderne Mikroprozessoren haben ihre gewaltige Leistungssteigerung unter anderem dem Einsatz von Caches zu verdanken. Sie überbrücken den großen Unterschied zwischen Prozessorgeschwindigkeit und Speicherzugriffszeit. Gerade die Verwendung von Caches macht aber die Berechnung der Laufzeit von Programmen schwierig. Deshalb galt lange Zeit die Empfehlung, in Echtzeitanwendungen keine Prozessoren mit Caches zu verwenden.

Je größer jedoch die Anwendungen werden, desto stärker werden die Anforderungen an die Rechenleistung der verwendeten Prozessoren. Moderne Microcontroller und digitale Signalprozessoren verfügen deshalb zunehmend über eingebaute Caches. Dadurch entstehen neue Anforderungen an die zur Entwicklung von Echtzeitsystemen verwendeten Übersetzer. Sie müssen Laufzeitgarantien für Systeme auf Prozessoren mit Caches abgeben, wobei die Ausführungszeit eines Speicherzugriffs sehr stark davon abhängt, ob der zugegriffene Speicherblock sich bereits im Cache befindet. Eine sichere, d. h. zur Abgabe einer Garantie verwendbare Annahme, wäre, „der Speicherblock ist nicht im Cache“. Dadurch bekäme man allerdings Laufzeitabschätzungen, die um Grössenordnungen schlechter als die tatsächlichen Laufzeiten wären. Um die vom Kunden verlangten Zeitgarantien geben zu können, müßte der Entwickler weit überdimensionierte Hardwareleistung vorsehen. Verläßliche Aussagen der Art, „der angesprochene Speicherblock ist bei jeder Ausführung im Cache“ würden die Laufzeitabschätzungen erheblich verbessern.

## 2 Statische Analyse

Die statische Analyse von Programmen ist eine Technik aus dem Übersetzerbau, mit der man Laufzeiteigenschaften von Programmen zur Übersetzungszeit annäherungsweise berechnet. Sie wurde zuerst für optimierende Übersetzer entwickelt, ist aber auch zur Unterstützung der Programmentwicklung, z. B. zum Testen verwendbar. Sie hat eine gut entwickelte theoretische Grundlage, auf der man die Korrektheit von Analysen nachweisen kann. Außerdem liefert diese theoretische Grundlage auch den Ansatz zur Generierung von Analysatoren. D. h. Analysatoren können aus Spezifikationen automatisch erzeugt werden.

## 3 Die statische Voraussage von Cache-Inhalten

Die Eigenschaft, daß ein Speicherblock bei jeder Ausführung einer Instruktion im Cache ist, ist eine der oben angesprochenen Laufzeiteigenschaften, die man annäherungsweise statisch berechnen kann. Dazu berechnet man für jeden Programmpunkt eine kompakte Darstellung aller dort möglicherweise auftretenden Cache-Inhalte. Diese Information ist *korrekt* in dem Sinn, daß jeder in der Darstellung enthaltene Speicherblock bei jeder Ausführung der Instruktion auch tatsächlich im Cache enthalten ist. Andererseits ist sie i. a. nicht *vollständig*, weil es weitere, nicht in der Darstellung enthaltene Speicherblöcke geben kann, die während der Ausführung im Cache sind. Die *Präzision* der Analyse bestimmt sich aus der Anzahl der von der Analyse vorausgesagten und der tatsächlich im Cache befindlichen Speicherblöcke.

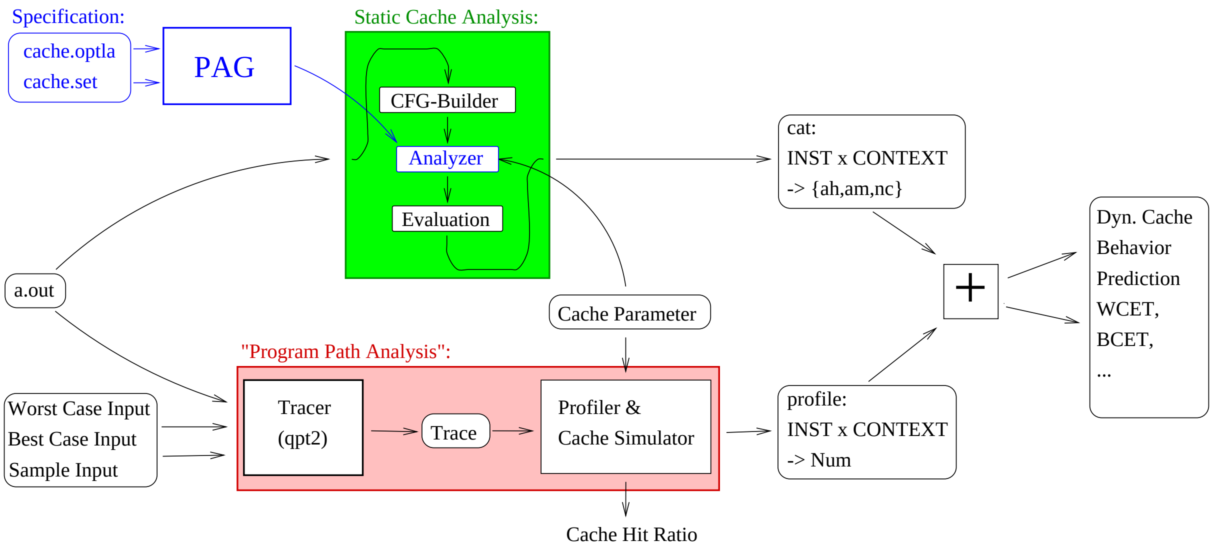


Abbildung 1: Die Struktur der Cacheanalyse.

Die von uns entwickelten Analysen sind

- wegen der systematischen Herleitung auf theoretischen Grundlagen **beweisbar korrekt**,
- wegen einer Darstellung der Menge der Cache-Inhalte, die wenig Informationen verliert, **sehr präzise**, erlauben also sehr genaue Voraussagen der tatsächlichen Cache-Inhalte,
- wegen der geschickten Darstellung der Cache-Inhalte und effizienter Algorithmen **schnell**,
- **generisch**, d. h. für viele verschiedene Cachearchitekturen anpaßbar,
- **automatisch erzeugt**.

## 4 Laufzeitabschätzungen

Die oben beschriebene Analyse liefert für jeden Programmpunkt eine (sichere) Untermenge der tatsächlich im Cache befindlichen Speicherblöcke. Eine zweite Analyse liefert eine (sichere) Obermenge. Diese Information ist nützlich, da man weiß, daß ein Speicherblock, der nicht in dieser Obermenge ist, auch nie im Cache sein wird. Beide Informationen zusammen kann man ausnutzen, Speicherzugriffe als *Cachehits* und *Cachemisses* zu klassifizieren. Mit dieser Klassifikation kann man die Abschätzungen für die beste und die schlechteste Ausführungszeit verbessern.

## 5 Experimentelle Ergebnisse

Die Cacheanalysatoren werden mit Hilfe des Programmanalysatorgenerators `PAG` aus prägnanten Spezifikationen generiert und arbeiten auf direkt ausführbaren Programmen im `a.out`-Format, so daß keine speziellen Übersetzer oder Linker notwendig sind. Die augenblickliche Implementierung unterstützt die SPARC-Architektur. Weitere Architekturen können durch die Bereitstellung entsprechender Frontends unterstützt werden.

Zehn verschiedene Benchmarkprogramme mit 135–5677 Instruktionen wurden für 25 verschiedene Cache-Konfigurationen von 128Bytes–20KBytes und verschiedene Cache-Assoziativitäten analysiert. Dabei konnte das Instruktionscache-Verhalten von im Durchschnitt 96.21% aller Instruktionen statisch vorhergesagt werden. Die daraus ermittelte Vorhersage der Trefferraten des Instruktionscaches weicht im Durchschnitt nur um 0.78 Prozentpunkte von den gemessenen Trefferraten ab.

## Literatur


[1] Christian Ferdinand, Florian Martin, und Reinhard Wilhelm. Applying Compiler Techniques to Cache Behavior Prediction. In *Proceedings of the ACM SIGPLAN Workshop on Language, Compiler and Tool Support for Real-Time Systems*, Seiten 37–46, Juni 1997.

[2] Christian Ferdinand, Florian Martin, und Reinhard Wilhelm. Cache Behavior Prediction by Abstract Interpretation. *Science of Computer Programming*, 1998. Ausgewählt für Sonderband SAS'96.

[3] Christian Ferdinand. Cache Behavior Prediction for Real-Time Systems. Dissertation, Universität des Saarlandes, September 1997.