Partitionierungs-Scheduling von heterogenen Tasksystemen auf Multiprozessoren in Automotive Steuergeräten*

Michael Deubzer, Jürgen Mottok
LaS³, University of Applied Science Regensburg,
Seybothstraße 2, 93053 Regensburg
{Michael.Deubzer;Juergen.Mottok}@hs-regensburg.de

Ulrich Margull
1 mal 1 Software GmbH,
Maxstraße 31, 90762 Fürth,
ulrich.margull@1mal1.com

Michael Niemetz, Gerhard Wirrer
Continental Automotive GmbH,
Siemensstraße 12, 93055 Regensburg
{Michael.Niemetz;Gerhard.Wirrer}@continental-corporation.com

Diese Arbeit befasst sich mit der statischen Aufteilung von heterogenen Tasksystemen auf SMP Multiprozessoren im Automotive Powertrain Bereich. Eine Besonderheit dieser Systeme ist eine Kombination aus kooperativer und preemptiver Verarbeitung von zeitgesteuerten und ereignisgesteuerten Tasks. Da die Aufteilung von Tasks ein Problem mit NP-Schwere ist, verwenden wir hierfür Partitionierungsheuristiken. Zur Validierung der Heuristiken präsentieren wir einen neuartigen Ansatz zur Echtzeitanalyse, basierend auf einer Multiprozessor-Schedulingsimulation.

Einleitung

Eingebettete Echtzeitsysteme im Automobilbereich setzten meist auf Singleprozessor Architekturen auf. Zusätzliche Rechenleistungsanforderungen wurden bisher maßgeblich durch die Erhöhung der Prozessortaktfrequenz realisiert. Neben der technisch möglichen Maximalfrequenz sind dieser Takterhöhung jedoch zunehmend Grenzen durch die erforderliche Limitierung der Wärmeemission und EMV-Belastung gesetzt. Multiprozessoren ermöglichen eine Erhöhung der Rechenleistung ohne diese Nachteile, erfordern aufgrund der Parallelität jedoch Anpassungen der Softwarearchitektur sowie geeignete Algorithmen zur Verteilung der Software auf die Prozessoren.

Das Scheduling von Multiprozessoren lässt sich in die zwei Bereiche globales Scheduling und Partitionierungs-Scheduling klassifizieren. Die Unterscheidung liegt darin, dass beim globalen Scheduling Tasks während der Laufzeit zwischen Prozessoren migrieren können, wohingegen beim Partitionierungs-Scheduling die Tasks beim Systemdesign den Prozessoren zugewiesen werden.

Globales Scheduling ermöglicht theoretisch höhere Auslastungen [Carpenter et al., 2004] erfordert jedoch zusätzliche Konsistenzmechanismen und hardwareseitige Unterstützung.

---

* Die vorliegende Arbeit ist gefördert durch das vom BMBF finanzierte Forschungsprojekt DynaS³, „Dynamische SW-Architekturen in Steuergeräten in Fahrzeugsystemen unter Berücksichtigung von Anforderungen zur Funktionalen Sicherheit“, Förderkennzeichen FHprofUnd2006 FKZ1752X07, weitere Informationen unter www.las3.de.

Partitionierungs-Scheduling hat den Vorteil, dass bekannte Schedulealgorithmen für Singleprozessor wie Earliest Deadline First (EDF) oder Rate Monotonic (RM) [Liu, 1973] genutzt werden können. Die Aufteilung der Tasks auf die Prozessoren ist hingegen ein Problem mit NP-Schwere, weswegen hierfür geeignete Heuristiken aufgestellt werden müssen. Untersuchungen zu periodischen Tasksystemen [Lopez et al., 2004] und sporadischen Tasksystemen [Fisher, 2007] zeigten, dass sich die maximale Auslastung U_{Sys} eines Systems auf

$$U_{Sys} = \left(\alpha^{-1}\mathrm{M} + 1\right) \Big/ (\alpha^{-1} + 1) \qquad (1)$$

beschränkt, wobei M die Anzahl der Prozessoren und $\alpha = \mathrm{e_i} / \mathrm{p_i}$ das maximale Taskgewicht, berechnet aus der Tasklaufzeit $\mathrm{e_i}$ und der Periode $\mathrm{p_i}$, darstellt.

In dieser Arbeit beschreiben wir das Partitionierungs-Scheduling als ersten möglichen Ansatz zur Überführung von bestehender Singleprozessor-Software auf Multiprozessoren. Hierzu stellen wir zunächst ein Taskmodell vor, welches speziell für den Automotive Powertrain Bereich zugeschnitten ist. Für den Partitionierungs-Scheduling-Ansatz beschreiben wir, wie bestehende Partitionierungsheuristiken auf dieses Taskmodell angewendet werden können und vergleichen es anschließend mittels eines systematischen Simulationsansatzes. Im letzten Abschnitt geben wir eine Zusammenfassung und einen Ausblick für weitere Untersuchungen.

Automotive Task System

Wie bereits erwähnt, zeichnen sich Automotive Systeme durch Heterogenität und Kooperation aus. Heterogenität bedeutet, dass Tasks ereignisgesteuert oder zeitgesteuert aktiviert werden können. Kooperation bedeutet, dass Tasks nur zu bestimmten Zeitpunkten in der Ausführung unterbrochen werden dürfen.

Das Taskset eines Automotive Systems $\tau = \{\mathrm{T_1}, ..., \mathrm{T_n}\}$ besteht somit aus zeitgesteuerten $\mathrm{T_i^Z}$ und ereignisgesteuerten $\mathrm{T_i^E}$ Tasks, die im Weiteren genauer beschrieben werden.

Ein ereignisgesteuerter Task ist durch $\mathrm{T_i^E} = \{\mathrm{p_i}, \mathrm{d_i}, \mathrm{E_i}, \mathrm{G_i}\}$ definiert, wobei $\mathrm{p_i}$ der minimalen Zeit zwischen zwei Aktivierungen, $\mathrm{d_i}$ der relativen Deadline und $\mathrm{E_i}$ der maximalen Ausführungszeit entspricht. Die Taskgruppe $\mathrm{G_i}$ dient zur Beschreibung der Kooperation. So können Tasks der gleichen Taskgruppe sich gegenseitig nur an kooperativen Schedule-Punkten unterbrechen. Für Tasks einer höheren Taskgruppe gilt das hingegen nicht. Diese können Tasks einer niedrigeren Gruppe sofort unterbrechen. Der Abstand zwischen zwei Schedule-Punkten wird als Taskphase bezeichnet. Die Summe aller Taskphasen ergibt die Ausführungszeit $\mathrm{E_i}$.

Ein zeitgesteuerter Task ist durch $\mathrm{T_i^Z} = \{\mathrm{o_i}, \mathrm{p_i}, \mathrm{d_i}, \mathrm{E_i}, \mathrm{G_i}\}$ definiert. Hier entspricht $\mathrm{p_i}$ dem konstanten Abstand zwischen zwei Aktivierungen. Durch die Verschiebung $\mathrm{o_i}$ der ersten

Aktivierung relativ zum Systemstartpunkt ist es möglich, die Aktivierung der Tasks zueinander zu versetzen und somit einen Lastausgleich vor dem eigentlichen Scheduling zu erreichen. Im Weiteren verwenden wir folgende Definitionen.

Definition 1 – Taskauslastung: Eine Task hat die Auslastung $\tilde{u}_i$, welche sich aus der maximalen Ausführungszeit E_i und dem minimalen Aktivierungsabstand p_i einer Task T_i errechnet.

$$\tilde{u}_i = E_i / p_i \qquad (2)$$

Definition 2 – Prozessorauslastung: Ein System besitzt M Prozessoren P_k mit einer Auslastung $\tilde{U}_k$, welche sich aus Summe der Auslastung $\tilde{u}_i$ aller Tasks T_i errechnet, die diesem Prozessor zugeteilt sind (Wir verwenden hierfür die Notation $T_i \in P_k$).

$$\tilde{U}_k = \sum_{i\,:\,T_i \in P_k} \tilde{u}_i \qquad (3)$$

Definition 3 – Freie Prozessorkapazität: Entsprechend der Prozessorauslastung besitzt ein Prozessor P_k eine freie Kapazität $\tilde{K}_k$.

$$\tilde{K}_k = 1 - \tilde{U}_k \qquad (4)$$

Heuristiken zur Partitionierung

In diesem Kapitel stellen wir Partitionierungsalgorithmen für das beschriebene Taskset vor. Dazu konzentrieren wir uns auf Verfahren basierend auf der Bin Packing Theorie [Garey & Johnson, 1979].

Beim Bin Packing werden Tasks T_i nach ihrer Auslastung $\tilde{u}_i$ den Prozessoren P_k zugeteilt. Als Auswahlkriterium für den Prozessor wird dessen freie Kapazität $\tilde{K}_k$ herangezogen.

Bei den Varianten FFD (First Fit Decreasing) und WFD (Worst Fit Decreasing) werden die Tasks T_i der Auslastung $\tilde{u}_i$ nach absteigend sortiert (Decreasing) und nacheinander dem Prozessor P_k zugewiesen.

- First Fit Decreasing: Für P_k muss gelten $k = min\left\{ j \mid \tilde{K}_j \geq \tilde{u}_i \right\}$
- Worst Fit Decreasing: Für P_k muss gelten $\tilde{K}_k = \max_{1 \leq j \leq M} (\tilde{K}_j)$

Simulationsansatz

Zum Vergleich der vorgestellten Partitionierungsheuristiken verwenden wir hier eine neuartige Analysetechnik, die für eine Vielzahl von Softwarearchitektur-Entscheidungen eingesetzt werden kann. Diese ermöglicht es, Sensitivitätsanalysen auf bestehenden und geplanten Systemen durchzuführen.

Grundlage der Analysetechnik ist eine diskrete ereignisorientierte Simulation. Hierbei wird das Verhalten von Hardware- und Softwarekomponenten durch Zustandsmodelle

nachgebildet. Entsprechend den Aktivierungsmustern des Tasksets werden Tasks aktiviert und durchlaufen die verschiedenen Zustandsübergänge wie z.B. Aktivierung, Unterbrechung, Terminierung. Die mittels der Simulation generierte Aufzeichnung enthält diese Übergänge mit Zeitstempel und ermöglicht es Performance- [Deubzer, 2008] und Echtzeitanalysen [König et al., 2009] durchzuführen.

Allerdings liefert die Art der Simulationsdurchführung nur Aussagen für ein bestimmtes System. In der Phase der Designentscheidungen von Softwareprojekten sind die Taskset-Parameter jedoch nicht genau bekannt.

Aus diesem Grund bedienen wir uns des Konzeptes der Monte Carlo Analyse, da hierdurch ein Teil des Taskset-Parameterraums untersucht wird. Dazu belegen wir Parameter des Tasksets mit Wahrscheinlichkeitsverteilungen und generieren entsprechend der Verteilungen eine Menge an Tasksets die anschließend simuliert werden.

Zur Simulation entwickelten wir ein Tool auf Basis von C++, welches es ermöglicht Multiprozessoren zu untersuchen. Da durch den Monte Carlo Ansatz die Anzahl der Simulationsdurchläufe stark ansteigt nutzen wir zur Durchführung ein Cluster-Computing Netzwerk mit dem High Throughput Computing System *Condor* der Wisconsin-Madison University [Condor].

Mittels der durch Buttazzo vorgestellten Metriken [Buttazzo, 2004] können die generierten Aufzeichnungen nun analysiert werden. Zur Analyse verwenden wir die Kenngröße Lateness, welche die Zeit zwischen der Fertigstellung einer Task bis zur Deadline darstellt. Ein negativer Latenesswert bedeutet, dass die Task die Deadline eingehalten hat, ein positiver Latenesswert entspricht einer Überschreitung der Deadline.

Zur Ermittlung der Kenngröße für das gesamte Taskset, wird die Instanz $T_{i,j}$ aller Tasks T_i eines Tasksets τ ermittelt, welche das Maximum der normierten Lateness L liefert [König et al., 2009]. Zur Normierung wird die Lateness einer Taskinstanz durch die entsprechende Deadline d_i dividiert. Das Maximum der normierten Lateness L errechnet sich demnach mittels Formel 5.

$$L = \max_{T_{i,j}} \left(\frac{l_{i,j}}{d_i} \right) \quad (5)$$

Zur Sensitivitätsanalyse werden die generierten Tasksets entsprechend ihrer Systemauslastung ($U_{SYS} = \sum_{1 \le k \le M} \tilde{U}_k$) in Gruppen einsortiert. Für jede Gruppe wird L berechnet sowie dessen Median und ein 99% Konfidenzintervall von $Q_{.99}$ und $Q_{0.01}$.

Fallstudie

In einer Fallstudie wurde ein existierendes Automotive Powertrain System bzgl. der Portierung auf ein Dual-Core System mittels den Partitionierungsheuristiken FFD und WFD untersucht. Die Parameter des existierenden Tasksets lagen dabei innerhalb der Grenzen aus Tabelle 1.

Task Typ	Deadline	Aktivierungsintervall	Anzahl Taskphasen	Offset	Task Gruppe	
T^Z	[0.3, …, 500]	[1, … ,1000]	[1,…,20]	[0, … ,100]	[1]	
T^E	[0.3, … ,5]	[1 ,…, 5]	[1,..,6]	-	[1,2]	

Tabelle 1: Grenzen der Taskset Parameter für Monte Carlo Generierung in ms.

Im Rahmen eines Monte Carlo Ansatzes wurden 10^4 Tasksets generiert, wobei die Laufzeit der Taskphasen (Abstand zwischen zwei Schedule-Punkten) nach einer Weibull-Verteilung ($\mathrm{Wei}(x) = (k/\lambda)(x/\lambda)^{k-1} e^{-(x/\lambda)k}$) innerhalb der Grenzen 0.1 - 0.3 ms variierten, wie in Abbildung 1 dargestellt. Da für die generierten Tasksets die Anzahl der Taskphasen konstant ist, deren Laufzeiten jedoch zufällig gezogen wird, resultierten somit unterschiedliche Tasklaufzeiten E_i und somit auch unterschiedliche Systemauslastungen für die Tasksets.

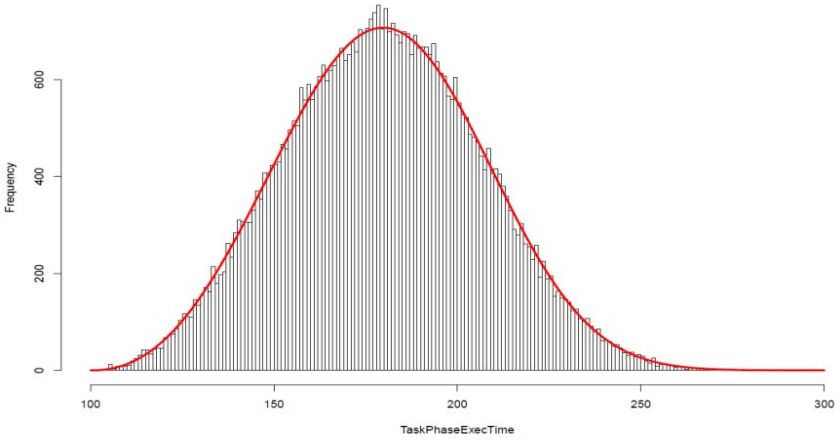

Abbildung 1: Histogramm der generierten Taskphasenlaufzeiten nach einer Weibull Verteilung (λ = 89.2, k =3.25) in µs.

Die Untersuchung der generierten Tasksets ist für RM Scheduling in Abbildung 2 dargestellt. Das Symbol stellt den Median dar und die äußeren Balken das 99% Konfidenzintervall von $Q_{.99}$ und $Q_{0.01}$. Die Abbildung zeigt, dass mittels einer FFD Heuristik und RM Scheduling Auslastungen bis zu 1,69 erreicht werden können. Damit wurde die Grenze für die maximale Systemauslastung nach (1) auch für die beschriebenen Systeme experimentell bestätigt ($\max_i(\tilde{u_i}) = 0,5$). Dagegen treten bei der WFD Heuristik schon im niedrigen Auslastungsbereich von 1,42 Deadlineverletzungen auf. Aus diesem Grund ist FFD für harte

Echtzeitsysteme besser geeignet. Im oberen Auslastungsbereich ab ~1,78 , erzielt WFD im Vergleich zu FFD durchschnittlich bessere Ergebnisse. Aufgrund der Deadlineverletzungen ist dieser Bereich jedoch nur für weiche Echtzeitsysteme geeignet.

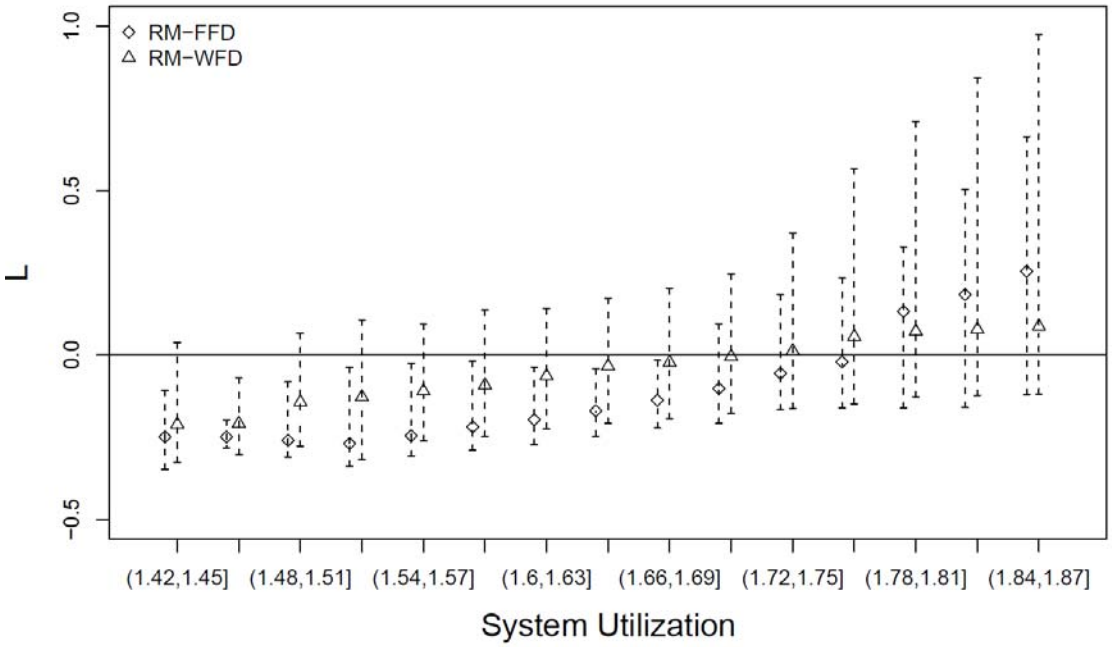

Abbildung 2: Vergleich der Partitionierungsheuristiken WFD und FFD mit dem Maximum der normierten Lateness L (median, $Q_{.99}$, $Q_{0.01}$) für generierte Tasksets, aufgetragen über die Systemauslastung.

Zusammenfassung

Partitionierung-Scheduling stellt den ersten Schritt der Migration von Singleprozessor Systemen auf Mehrprozessor Systemen dar. Die bestehenden Algorithmen, basierend auf Bin-Packing Heuristiken können auch im Automobilbereich eingesetzt werden, wie exemplarisch gezeigt wurde. Allerdings ist für harte Echtzeitsysteme die Leistungssteigerung durch Dual-core eingeschränkt. Für die untersuchten Tasksets beträgt diese 1,69 bei einem Dual-Core System mit FFD Partitionierung und RM Scheduling.

Eine Erweiterung dazu stellen iterative Partitionierungsalgorithmen dar, die z.B. durch genetische Algorithmen eine Aufteilung ermitteln, die bessere Latenesswerte produzieren. Erste Untersuchungen zeigen hier vielversprechende Ergebnisse und sind das Ziel weiterer Forschungsarbeiten.

Referenzen

[Carpenter et al., 2004] Carpenter, J. und Funk, S. und Holman, P. und Srinivasan, A. und Anderson, J. und Baruah, S. (2004): A categorization of real-time multiprocessor scheduling problems and algorithms. Handbook on Scheduling Algorithms, Methods, and Models.

[Deubzer et al., 2008] Deubzer, M. und Mottok, J. und Flug, C. und Zeitler, T. (2008): Profiling Performance Analyse von Embedded Real-Time System Architekturen, 1th ESE Congress

[König et al., 2009] König, F. und Boers, D. und Slomka, F. und Margull, U. und Niemetz, M. und Wirrer, G. (2009), Application Specific Performance Indicators for Quantitative Evaluation of the Timing Behavior for Embedded Real-Time Systems, Proceedings of the 12th conference on Design, automation and test in Europe.

[Condor] Condor Project®, (http://www.cs.wisc.edu/condor/)

[Fisher, 2007] Fisher, N. W. (2007): The multiprocessor real-time scheduling of general task systems. PhD thesis, University of North Carolina.

[Garey & Johnson, 1979] Garey, M. und Johnson D. (1979): Computers and Intractability. New York: W.H.Freman.

[Liu, 1973] Liu, C. L. und Layland, J. W. (1973), “Scheduling algorithms for multiprogramming in a hard real-time environment,” Journal of the Association of Computing Machinery, vol. 20.

[Lopez et al., 2004] Lopez, J. M. und Diaz, J. L., und Garcia, D. F. (2004): Utilization bounds for edf scheduling on real-time multiprocessor systems. Real-Time Systems, 28:39-68.
[Schöneburg et al., 1994] Schöneburg, E. und Heinzmann F. und Feddersen S. (1994): Genetische Algorithmen und Evolutionsstrategien, Addison-Wesley Verlag.

[Buttazzo,2004] Buttazzo, Giorgio C. (2004), Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications, Springer, New York.

Autoren

Dipl.-Ing. (FH) Michael Deubzer ist Doktorand im FHprofUnd-Forschungsprojekt DynaS³ am LaS³ der Regensburger Hochschule für Angewandte Wissenschaften. Der Schwerpunkt seiner Forschungsarbeit liegt in der Untersuchung von Softwarearchitekturen zur parallelen Verarbeitung in Eingebetteten Echtzeitsystemen.

Prof. Dr. Jürgen Mottok lehrt Software Engineering, Programmiersprachen, Betriebssysteme und Safety an der Regensburger Hochschule für Angewandte Wissenschaften. Er leitet das Software-Engineering-Labor „Laboratory for Safe and Secure Systems" (www.las3.de).

Dr. Michael Niemetz arbeitet als Experte für Softwarearchitektur bei der Continental Automotive GmbH in Regensburg im Bereich Powertrain Engine Systems Engineering.

Dipl.-Ing. (TU) Gerhard Wirrer leitet die Abteilung Funktions- und Softwarearchitektur bei der Continental Automotive GmbH im Bereich Powertrain Engine Systems Engineering.

Dr. Ulrich Margull ist Geschäftsführer der 1 mal 1 Software GmbH. Er berät Firmen in den Bereichen Softwarearchitektur und Echtzeitverhalten.