Zustellerkonzepte in Mehrkern-Systemen

Eine Ausarbeitung im Rahmen des KvBK-Seminars

Moritz König
Friedrich-Alexander-Universität Erlangen-Nürnberg
moritz.koenig@fau.de

ZUSAMMENFASSUNG

Diese Ausarbeitung befasst sich mit der Bearbeitung von nicht periodischen Aufgaben in Mehrkern-Echtzeitsystemen, die in realen Echtzeitsystemen eine wichtige Rolle spielen, durch ihre Unvorhersehbarkeit jedoch schwieriger zu behandeln sind als periodische Aufgaben. Der Fokus der vorgestellten Optionen liegt dabei auf der Optimierung des Antwortverhaltens von aperiodischen Aufgaben besonders für Systeme mit partitionierter Einplanung mittels des Earliest-Deadline-First-Algorithmus und mit Hilfe des Total-Bandwidth-Server-Algorithmus.

1. EINLEITUNG

Bei der Entwicklung von Echtzeitsystemen lag der Fokus über längere Zeit auf der Einplanung periodischer Aufgaben, durch die sich auch viele Probleme modellieren lassen.

Aber in realen Echtzeitsystemen spielen auch sporadische und aperiodische Aufgaben eine große Rolle. So sollen zum Beispiel in einigen Systemen Nutzereingaben zu jeder Zeit möglich sein oder eine Fehlerbehandlung bei Materialfehlern eines gesteuerten Gerätes erfolgen. Diese sind jedoch schwieriger zu behandeln als periodische Aufgaben, da ihr Auftreten nicht vorhersehbar ist.

Vermeintlich einfachstes Vorgehen, die nicht periodischen Aufgaben einzuplanen, ohne die rechtzeitige Behandlung der periodischen zu gefährden, ist es, Zeitslots, die nach der Einplanung von periodischen Aufgaben noch frei sind, den nicht periodischen zuzuteilen. Dieses Vorgehen hat jedoch ein schlechtes Antwortverhalten zur Folge.

Es ergibt sich also eine neue Zielsetzung: die Minimierung der Antwortzeiten aperiodischer Aufgaben. Dabei soll jedoch nicht die Möglichkeit, die Fristen periodischer Aufgaben garantiert einhalten zu können, gefährdet werden.

Übliche Lösung für das Problem der Behandlung nicht periodischer Aufgaben ist es, einen Zusteller (engl. server) einzuführen, der mit einem zugeteilten Budget nicht periodische Aufgaben bearbeitet und genauso wie periodische Aufgaben eingeplant wird.

Für Mehr-Prozessor-Systeme wurden jedoch, obwohl deren Nutzung üblich geworden ist, nur wenige Zustellerverfahren entwickelt, die ihren Fokus auf die Optimierung des Antwortverhaltens legen.

Im Folgenden wird zuerst der bisherige Stand der Wissenschaft erläutert. Im Anschluss wird das Systemmodell spezifiziert. Abschließend folgt die Beschreibung von Einplanungsverfahren für nicht periodische Aufgaben in Mehrkern-Echtzeitsystemen, mit denen das Antwortverhalten optimiert werden soll. Dabei befasst sich das erste Verfahren mit Aufgaben, die auf beliebigen Prozessoren laufen können, und das andere mit solchen, die auf dem Prozessor bearbeitet werden müssen, auf dem sie ankommen.

2. STAND DER WISSENSCHAFT

Andersson, Abdelzaher und Jonsson hatten die Idee, aperiodischen Aufgaben eine Frist zuzuteilen [7], wie den periodischen auch, wodurch sich die garantierte Echtzeitlauffähigkeit sowohl für globale als auch für partitionierte Einplanung verbessert. Bei globaler Einplanung kann für eine Auslastung von weniger als 50% garantiert werden, dass alle Fristen eingehalten werden [1], bei partitionierter für eine Auslastung von weniger als 31% [2].

Baruah und Lipari entwickelten einen auf dem Total-Bandwidth-Server (TBS), der zu Beginn des Kapitels 4 erläutert wird, und dem Earliest-Deadline-First (EDF) basierenden Algorithmus für Mehr-Prozessor-Echtzeitsysteme. Garantie der Echtzeitlauffähigkeit kann durch einen Planbarkeitstest gegeben werden [5].

Diese Arbeiten sind sehr wichtig, da eine Garantie für Echtzeitlauffähigkeit damit auch auf Mehrkern-Systemen gegeben werden kann. Das Ziel dieser Arbeiten war es jedoch somit eher, die zeitlichen Bedingungen einzuhalten, nicht das Antwortverhalten der aperiodischen Aufgaben zu verbessern.

Banús, Arenas, und Labarta entwickelten den von Davis und Wellings erdachten [6] Dual-Priority-Algorithmus weiter, der das Antwortverhalten in Einkern-Systemen mit festen Prioritäten verbessert, ohne die Fristen von periodischen Aufgaben zu verletzen [3].

Außer den in den folgenden Kapiteln erläuternden Verfahren existieren allerdings noch kaum Verfahren für partitionierte Einplanung auf EDF-Basis, die Antwortzeiten minimieren, obwohl partitioniertes Einplanen wegen Simplizität, Synchronisation und aus anderen Gründen in der Praxis oft bevorzugt wird. Außerdem kann EDF eine bessere Auslastung erreichen als statisch priorisierte Algorithmen [7].

3. SYSTEM-ABSTRAKTION

Das in den folgenden Kapiteln verwendete Modell ist ein Mehrkern-Echtzeitsystem bestehend aus M Prozessoren: P_1, $P_2, ..., P_M$. Die Aufgaben sind dabei unter den Prozessoren aufgeteilt.

Das System hat eine Menge von n periodischen Aufgaben $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ zu bearbeiten, wobei die i-te periodische Aufgabe $\tau_i(C_i, T_i)$ eine Worst-Case-Execution-Time (WCET) von C_i und eine Periode von T_i hat. Eine Aufgabe τ_i hat

dabei eine Prozessorauslastung von $U_i = \frac{C_i}{T_i}$, die Gesamtauslastung beträgt $U(\Gamma) = \sum_{i=1}^{n} U_i$.

Der j-te Arbeitsauftrag von $\tau_i : \tau_{i,j}$ tritt dabei zur Zeit $r_{i,j}$ auf und hat eine der Auftrittszeit des (j+1)-ten Auftrages entsprechende Frist $d_{i,j} = r_{i,j} + T_i$.

Im verwendeten Modell treten nicht periodische Aufgaben sequentiell auf. Das heißt, die k-te nicht periodische Aufgabe $\alpha_k(a_k, E_k)$ hat dabei eine Ankunftszeit von a_k und eine WCET von E_k mit $\forall k : a_k < a_{k+1}$. Die Antwortzeit von α_k beträgt $R_k = f_k - a_k$, wobei f_k die Zeit der Fertigstellung von α_k darstellt. Die übrige Ausführungszeit des Arbeitsauftrags $\tau_{i,j}$ zur Zeit t wird mit $\tilde{c}_{i,j}(t)$ bezeichnet. Aus der Bedienrate μ des Zustellers und der durchschnittlichen Ankunftsrate λ von nicht periodischen Aufgaben ergibt sich die nicht periodische Auslastung $U(\alpha) = \frac{\lambda}{\mu}$.

Jede periodische Aufgabe ist genau einem Prozessor zugeordnet und wird über EDF eingeplant. Alle Aufgaben sind präemptiv und unabhängig. Jeder Prozessor kann nur eine Aufgabe gleichzeitig ausführen und jede Aufgabe wird auf höchstens einem Prozessor ausgeführt. Gesamtauslastung eines Prozessors P_x ist dabei $U(P_x)$.

Negative Auswirkungen von Interprozessorkommunikation und Migration, beispielsweise durch Cache-Effekte, sowie andere Anomalien werden im Folgenden nicht beachtet.

4. EINPLANUNGSVERFAHREN

Dieser Abschnitt beschäftigt sich mit der Optimierung des Antwortverhaltens von aperiodischen Aufgaben in Mehrkern-Echtzeitsystemen mit partitionierter EDF-Einplanung.

Zu Beginn erfolgt eine Erläuterung des TBS-Algorithmus, worauf die Beschreibung eines Verfahrens folgt, das die Antwortzeiten für aperiodische Aufgaben minimiert, die auf jedem Prozessor ausgeführt werden können. Abschließend wird eine Möglichkeit dargelegt, die Antwortzeit zu minimieren, wenn aperiodische Aufgaben auf dem Prozessor ausgeführt werden müssen, auf dem sie eintreffen.

4.1 Total-Bandwidth-Server

Die Idee des TBS-Algorithmus ist es, aperiodischen Aufgaben eine virtuelle Frist zu setzen, um diese dann genauso wie periodische Aufgaben über EDF einzuplanen. Um die Antwortzeiten zu minimieren wird die virtuelle Frist so früh wie möglich gewählt. Die Berechnung dieser basiert auf der Bandbreite beziehungsweise der Auslastung des Zustellers. Wenn der Prozessor P_x einen Zusteller τ_x^{srv} mit einer Bandbreite von U_x^{srv} hat, so berechnet sich die virtuelle Frist für eine aperiodische Aufgabe $\alpha_{x,k}$ mit Erscheinungszeit $a_{x,k}$ und WCET $E_{x,k}$ auf folgende Weise iterativ: [7]

$v_{x,0} := 0$

Für $k \neq 0$:

$$v_{x,k} := max\{a_{x,k}, v_{x,k-1}\} + \frac{E_{x,k}}{U_x^{srv}} \quad (1)$$

Abbildung 1 zeigt ein Beispiel eines Ablaufplans für einen Prozessor P_x mit EDF-Einplanung mit Hilfe des TBS-Algorithmus. Auf dem Prozessor sind zwei periodische Aufgaben auszuführen: $\tau_1(3, 6)$ und $\tau_2(2, 8)$. Somit ergibt sich eine Hyperperiode der Länge 24 und die Bandbreite des Zustellers von $U_x^{srv} = 1 - U(\Gamma) = 1 - (\frac{3}{6} + \frac{2}{8}) = 0.25$. Für $\alpha_{x,1}(2, 2)$ ergibt sich also eine virtuelle Frist von $v_{x,1} = max\{a_{x,1}, v_{x,0}\} + \frac{E_{x,1}}{U_x^{srv}} = 2 + \frac{2}{0.25} = 10$, was dazu führt, dass $\tau_{1,1}$ beendet und $\tau_{2,1}$ zuerst ausgeführt wird, da dieser Arbeitsauftrag eine Frist von $d_{2,1} = 8$ hat. Analog errechnet sich die Frist von $\alpha_{x,2}(7, 1)$: $v_{x,2} = max\{a_{x,2}, v_{x,1}\} + \frac{E_{x,2}}{U_x^{srv}} = 10 + \frac{1}{0.25} = 14$. Somit wird auch diese Aufgabe nicht sofort ausgeführt, da $\tau_{1,2}$ die höhere Priorität hat. Dem zuletzt auftretenden $\alpha_{x,3}(17, 2)$ kann schließlich $v_{x,3} = max\{a_{x,3}, v_{x,2}\} + \frac{E_{x,3}}{U_x^{srv}} = 17 + \frac{2}{0.25} = 25$ zugeordnet werden, sodass dieses ebenfalls nicht sofort ausgeführt wird, sondern auf Fertigstellung der höher priorisierten $\tau_{2,3}$ und $\tau_{1,4}$ warten muss.

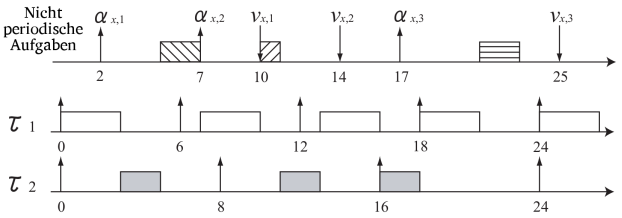

Figure 1: Ein Beispiel für Einplanung mit TBS [7]

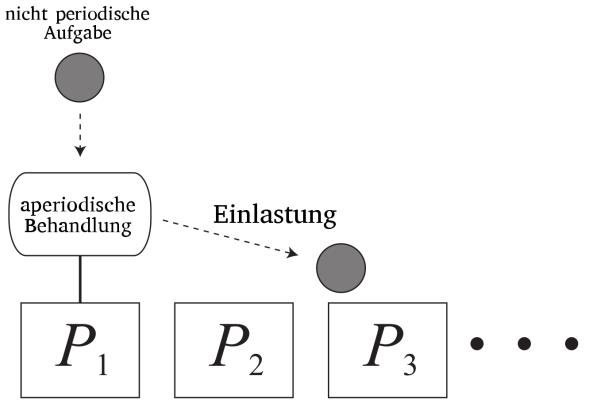

Figure 2: Prinzip des Einlastungsverfahrens [7]

4.2 Einlastungsverfahren

Die Idee des Einlastungsverfahrens, das die Antwortzeiten von aperiodischen Aufgaben, die auf beliebigen Prozessoren in Multikern-Echtzeitsystemen laufen können, minimiert, ist, wie Abbildung 2 zeigt, eher simpel: nicht periodische Aufgaben werden auf dem Prozessor eingeplant und später eingelastet, auf dem die Aufgabe vorraussichtlich die geringste Antwortzeit hat. In diesem Fall kommt die Aufgabe auf Prozessor P_1 an, hat aber die früheste virtuelle Frist auf Prozessor P_3 und wird somit dort eingelastet.

Die tatsächlich schlechtest mögliche Antwortzeit (engl. Worst-Case-Response-Time, WCRT) einer Aufgabe lässt sich zwar durch den TBS-Algorithmus schwer bestimmen, da die Reihenfolge der Aufgaben dynamisch ist, aber die virtuelle Frist aus obigem Abschnitt kann als WCRT angenommen werden, da die Aufgaben mit ihr als Deadline eingeplant werden, auch wenn sich die tatsächliche WCRT unterhalb dieser Frist befinden kann. Eine nicht periodische Aufgabe wird also auf dem Prozessor eingeplant und eingelastet, auf dem die virtuelle Frist am frühesten ist, bzw. auf dem Prozessor mit dem geringsten $v_{x,k}$ aus Gleichung (1) [7].

Die Tests von Kato und Yamasaki belegen, dass das Ein-

lastungsverfahren trotz seiner Einfachheit einen sehr positiven Effekt auf das Antwortverhalten der aperiodischen Aufgaben hat [7].

4.3 Migrationsverfahren

Das Migrationsverfahren beschäftigt sich mit der Optimierung des Antwortverhaltens von aperiodischen Aufgaben, die in einem Mehrkern-Echtzeitsystem auf dem Prozessor ausgeführt werden müssen, auf dem sie aufgetreten sind. Die Idee hierbei ist es, wie in Abbildung 3 veranschaulicht, periodische Aufgaben temporär auf andere Prozessoren zu migrieren. In diesem Fall wartet eine nicht periodische Aufgabe auf Prozessor P_1, weil dort zwei höher priorisierte periodische Arbeitsaufträge eingeplant sind. Diese zwei können hier jedoch ohne Fristverletzung auf P_2 und P_3 migriert werden, wodurch sich die Antwortzeit der nicht periodischen Aufgabe reduziert.

Die genaue Vorgehensweise lässt sich durch folgenden Algorithmus beschreiben. Wenn zu beliebiger Zeit t eine beliebige aperiodische Aufgabe $\alpha_{x,k}$ mit Ankunftszeit $a_{x,k}$ und geschätzter WCET $E_{x,k}$ auf Prozessor P_x auftritt, der einen Zusteller τ_x^{srv} mit Bandbreite U_x^{srv}, sowie mit einer virtuellen Frist $v_{x,k}$, hat, wendet das Verfahren die folgenden Schritte an:

1. Suche einen periodischen Arbeitsauftrag $\tau_{i,j}$ mit der frühesten Frist, der in seiner Periode noch nicht migriert wurde. Falls kein solcher Arbeitsauftrag existiert, endet der Algorithmus.

2. Suche einen Prozessor P_y, der die folgende Ungleichung (2) erfüllt, bei der $v_{y,l}$ die virtuelle Frist der letzten nicht periodischen Aufgabe $\alpha_{y,l}$ ist, die auf P_y aufgetreten ist. $\tilde{c}_{i,j}(t)$ ist dabei die restliche Ausführungszeit von $\tau_{i,j}$.

$$d_{i,j} \geq max\{t, v_{y,l}\} + \frac{\tilde{c}_{i,j}(t)}{U_y^{srv}} \tag{2}$$

3. Migriere $\tau_{i,j}$ zu P_y, wobei der Arbeitsauftrag als (l+1)-te nicht periodische Aufgabe auf P_y mit virtueller Frist $v_{y,l+1}$ aus folgender Gleichung (3) behandelt wird. Zu beachten ist, dass die nächste tatsächliche nicht periodische Aufgabe, die auf P_y ankommt, damit mit $\alpha_{y,l+2}$ bezeichnet wird.

$$v_{y,l+1} = max\{t, v_{y,l}\} + \frac{\tilde{c}_{i,j}(t)}{U_y^{srv}} \tag{3}$$

4. Weise $\alpha_{x,k}$ die verbesserte virtuelle Frist $v'_{x,k}$ aus folgender Gleichung (4) zu.

$$v'_{x,k} = max\{t, v_{x,k-1}\} + \frac{E_{x,k}}{U_x^{srv} + \frac{\tilde{c}_{i,j}(t)}{T_i}} \tag{4}$$

5. Migriere τ_i zur Zeit $r_{i,j+1}$ für ihren nächsten Arbeitsauftrag $\tau_{i,j+1}$ zurück zu P_x.

Anzumerken ist, dass es zwar eigentlich nicht das Beste ist, den Arbeitsauftrag $\tau_{i,j}$ als nicht periodische Aufgabe zu behandeln und ihm die virtuelle Frist $v_{y,l+1}$ zuzuweisen, da periodische Arbeitsaufträge nicht so schnell wie möglich fertiggestellt sein sollen und so eventuell unnötig Bandbreite verschwendet wird. Das Zuweisen der tatsächlichen Frist $d_{i,j}$ macht jedoch die Planbarkeitsanalyse erheblich komplizierter. Die Verhinderung von Mehrfachmigration wird aus praktischen Gründen im ersten Schritt verhindert [7].

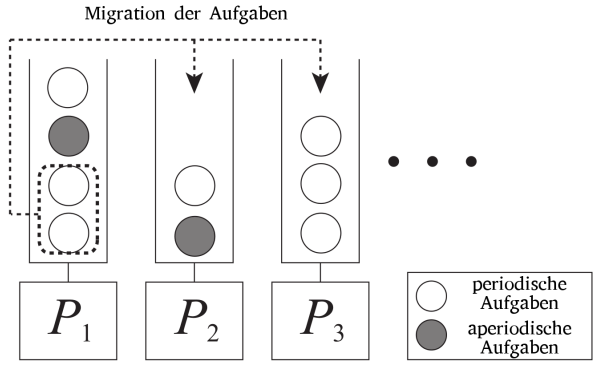

Figure 3: Prinzip des Migrationsverfahrens [7]

Abbildung 4 zeigt ein Beispiel eines Ablaufplans für zwei Prozessoren P_x und P_y mit oben erklärtem Migrationsverfahren. Auf P_x sind zwei periodische Aufgaben auszuführen: $\tau_1(3,6)$ und $\tau_2(2,8)$. Somit ergibt sich die Bandbreite des Zustellers von $U_x^{srv} = 1 - U(\Gamma) = 1 - (\frac{3}{6} + \frac{2}{8}) = 0.25$. Zusätzlich werden $\tau_3(1,4)$ und $\tau_4(5,10)$ auf P_y eingeplant, was eine Bandbreite des Zustellers von $U_y^{srv} = 0.25 = U_x^{srv}$ bedeutet. Um die Erklärung einfach zu halten, trifft in diesem Beispiel keine nicht periodische Aufgabe auf P_y ein. Da die $\alpha_{x,k}$, τ_1 und τ_2 denen aus Abbildung 1 entsprechen, wäre die virtuelle Frist für $\alpha_{x,1}(2,2)$ ohne Migration $v_{x,1} = 10$. Mit Anwendung des Verfahrens hingegen kann $\tau_{1,1}$ auf P_y migriert werden, da dies Ungleichung (2) erfüllt. Der Arbeitsauftrag erhält dabei nach Gleichung (3) eine virtuelle Frist von $v_{y,1} = t + \frac{\tilde{c}_{1,1}(t)}{U_y^{srv}} = 2 + \frac{1}{0.25} = 6$. In der Folge erhält $\alpha_{x,1}$ nach Gleichung (4) die verbesserte virtuelle Frist $v'_{x,1} = max\{t, v_{x,0}\} + \frac{E_{x,1}}{U_x^{srv} + \frac{\tilde{c}_{1,1}(t)}{T_1}} = 2 + \frac{2}{0.25 + \frac{1}{6}} = 6.8$, was dazu führt, dass $\alpha_{x,1}$ sofort ausgeführt werden kann, da es auf P_x damit die höchste Priorität besitzt. Bei Eintreten von $\alpha_{x,2}(7,1)$ hingegen kann τ_1 nicht temporär auf P_y migriert werden, da Ungleichung (2) nicht erfüllt ist: $d_{1,2} = 12 \ngeq max\{t, v_{y,2}\} + \frac{\tilde{c}_{1,2}(t)}{U_y^{srv}} = 7 + \frac{2}{0.25} = 15$. Deshalb kann $\alpha_{x,2}$ auch keine frühere Frist erhalten. Bei dem zuletzt auftretenden $\alpha_{x,3}(17,2)$ kann schließlich τ_2 temporär auf P_y migriert werden, da hier die Ungleichung (2) wieder erfüllt wird. $\tau_{2,3}$ erhält dabei nach Gleichung (3) eine virtuelle Frist von $v_{y,2} = max\{t, v_{y,1}\} + \frac{\tilde{c}_{2,3}(t)}{U_y^{srv}} = 17 + \frac{1}{0.25} = 21$ und folglich $\alpha_{x,3}$ nach Gleichung (4) die verbesserte virtuelle Frist $v'_{x,3} = max\{t, v_{x,2}\} + \frac{E_{x,3}}{U_x^{srv} + \frac{\tilde{c}_{2,3}(t)}{T_2}} = 17 + \frac{2}{0.25 + \frac{1}{8}} = 22.3$ und kann damit als höchst priorisierter Prozess auf P_x sofort ausgeführt werden.

4.3.1 Antwortverhalten

Das oben beschriebene Migrationsverfahren bietet gleich zwei Vorteile für das Antwortverhalten von aperiodischen Aufgaben.

Der erste Vorteil ist, dass aperiodische Aufgaben möglicherweise früher eingelastet werden können, da höher priorisierte periodische Arbeitsaufträge migriert sein können, und somit nicht mehr vorher auf dem entsprechenden Prozessor ein-

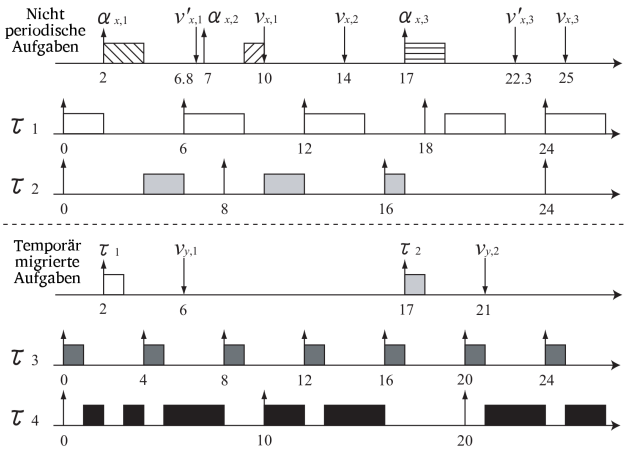

Figure 4: Ein Beispiel für das Migrationsverfahren [7]

gelastet werden können. Die Antwortzeit kann sich also verbessern, ohne dass die virtuelle Frist früher ist.

Der zweite Vorteil ist, dass die temporäre Migration von periodischen Arbeitsaufträgen auch die Bandbreite des Zustellers eines Prozessors temporär erhöht, weil dieser den Arbeitsauftrag nicht mehr bearbeiten muss. Dies hat, wie man an Gleichung (4) sehen kann, zur Folge, dass die virtuellen Fristen kleiner werden, was wiederum die frühere Einlastung und somit Fertigstellung bewirkt.

Anzumerken ist, dass die Migration von einem periodischen Arbeitsauftrag auf einen Prozessor das Antwortverhalten der nicht periodischen Aufgaben, die bereits zuvor auf diesem Prozessor eingegangen sind nicht beeinträchtigt, was an Ungleichung (2) zu sehen ist. Später ankommende Aufgaben können jedoch ein schlechteres Antwortverhalten aufweisen, da sie erst nach Fertigstellung des migrierten Arbeitsauftrages eingelastet werden können.

Das Verfahren versucht jedoch immer, wenn es möglich ist, periodische Arbeitsaufträge zu migrieren, was insgesamt eine Verbesserung des Antwortverhaltens erwarten lässt. In der Tat haben die Tests von Kato und Yamasaki diese Annahme auch bestätigt, wobei der positive Effekt geringer ausfällt, als bei dem Einlastungsverfahren [7].

4.3.2 Planbarkeit

Nach der TBS-Theorie sind der migrierte Arbeitsauftrag $\tau_{i,j}$ und die ursprünglichen Arbeitsaufträge des Prozessors garantiert planbar, da die Zuweisung einer virtuellen Frist, die kleiner als die tatsächliche ist, die Planbarkeit des EDF-Algorithmus nicht beeinflusst [9] [8]. Deshalb sind auch die ursprünglichen Arbeitsaufträge noch garantiert planbar, was die große Tragfähigkeit des EDF-Algorithmus bedingt [4]. Die zeitlichen Anforderungen werden also nicht verletzt.

4.3.3 Wahl des Zielprozessors

Der folgende Abschnitt widmet sich der Wahl des Zielprozessors in Schritt zwei des Algorithmus, falls sich dort mehrere Möglichkeiten ergeben. Da das Auffinden der tatsächlich besten Lösung ein NP-schweres Problem ist, ist es sinnvoll eine Heuristik anzuwenden, wobei sich die folgenden anbieten:

- Die First-Fit-Heuristik migriert den periodischen Arbeitsauftrag, wie der Name sagt, auf den ersten Prozessor, der den Anforderungen aus Ungleichung (2) genügt.
- Die Best-Fit-Heuristik migriert $\tau_{i,j}$ zu dem Prozessor, bei dem die Differenz $|d_{i,j} - v_{y,l+1}|$ minimal ist und Ungleichung (2) erfüllt.
- Die Worst-Fit-Heuristik maximiert entsprechend diese Differenz

Laut den Tests von Kato und Yamasaki hat sich die Worst-Fit-Heuristik als am effektivsten herausgestellt [7].

5. SCHLUSS

Diese Ausarbeitung erläutert Einplanungsstrategien für Mehrkern-Echtzeitsysteme mit Fokus auf die Antwortzeitminimierung von aperiodischen Aufgaben bei einer partitionierten EDF-Einplanung.

Das Verfahren für Aufgaben, die auf beliebigen Prozessoren bearbeitet werden können, zeichnet sich dabei als sehr simpel und effektiv aus. Das Einlastungsverfahren muss schließlich nur das Minimum einer kleinen endlichen Menge von Ungleichungen bestimmen.

Auch das Migrationsverfahren, bei dem periodische Aufgaben auf einem Prozessor temporär ausgelagert werden, wurde beschrieben. Dieses eignet sich für Aufgaben, die auf dem Prozessor ausgeführt werden müssen, auf dem sie aufgetreten sind, und hat sich ebenfalls als effektiv herausgestellt.

In zukünftigen Arbeiten wird sich wohl noch damit auseinandergesetzt werden, wie sehr oben erwähnte Anomalien in die Antwortzeiten eingehen, aber auch mit der Planbarkeitsanalyse für den Fall, dass migrierten Arbeitsaufträgen ihre tatsächliche Frist zugewiesen wird.

6. LITERATUR

[1] B. Andersson, T. Abdelzaher, and J. Jonsson. Global Priority-Driven Aperiodic Scheduling on Multiprocessors. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2003.

[2] B. Andersson, T. Abdelzaher, and J. Jonsson. Partitioned Aperiodic Scheduling on Multiprocessors. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2003.

[3] J. Banús, A. Arenas, and J. Labarta. Dual Priority Algorithm to Schedule Real-Time Tasks in a Shared Memory Multiprocessor. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2003.

[4] S. Baruah and A. Burns. Sustainable Scheduling Analysis. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 159–168. IEEE, 2006.

[5] S. Baruah and G. Lipari. A multiprocessor implementation of the Total Bandwidth Server. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, pages 40–47. IEEE, 2004.

[6] R. Davis and A. Wellings. Dual Priority Scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 100–109. IEEE, 1995.

[7] S. Kato and N. Yamasaki. Scheduling Aperiodic Tasks using Total Bandwidth Server on Multiprocessors. In

2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, volume 1, pages 82–89. IEEE, 2008.

[8] M. Spuri and G. Buttazzo. Scheduling Aperiodic Tasks in Dynamic Priority Systems. *Real-Time Systems*, 10(2):179–210, 1996.

[9] M. Spuri and G. C. Buttazzo. Efficient Aperiodic Service under Earliest Deadline Scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 2–11. IEEE, 1994.