A taxonomy of real time scheduling theory feasibility tests

Frank Singhoff
LISyC/University of Brest/UEB

11 septembre 2011

Table des matières

1 Introduction

This report presents all feasibility tests that we have selected to be implemented into the Cheddar framework. We mostly assume systems that are composed of periodic tasks running on mono-processor architectures. See [GRS96, KRP$^+$94] for a more detailed review of feasibility tests.

2 Some definitions

Some definitions that are handled in the rest of this report.

Définition 1 *Non concrete task*
In this report, a non concrete (periodic) task is defined by parameters P_i, D_i and C_i.

Définition 2 *Concrete task*
In this report, a concrete (periodic) task is defined by parameters P_i, D_i, C_i. and S_i.

Définition 3 *Synchronous concrete task set*
A synchronous concrete task set is a set of concrete tasks with $\forall i : S_i = k$ where k is a critical instant.

Définition 4 *Asynchronous concrete task set*
A concrete task set is a set a tasks with no critical instant.

Définition 5 *Request on deadline tasks*
A request on deadline task is a task with $D_i = P_i$.

3 Feasibility tests taxonomy

4 Verification based on exhaustive simulations

4.1 S1 test : any real time schedulers

1. **How to compute :**

$$[0, LCM(\forall i : P_i)]$$

2. **References :** [RRC02].
3. **Assumptions :**
   - Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   - $LCM(\forall i : P_i)$ is the last common multiplier of all period values of the task set.

4.2 S2 test : any real time schedulers

1. **How to compute :**

$$[0, LCM(\forall i : P_i) + 2 \cdot max(\forall i : S_i)]$$

2. **References :** [RRC02].
3. **Assumptions :**
   – Independent periodic tasks. Concrete and asynchronous task set.
4. **Comments :**
   – $LCM(\forall i : P_i)$, the last common multiplier of all period values of the task set.

5 Tests based on processor utilization factor

5.1 C1 test : preemptive Rate Monotonic

1. **How to compute :**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq B$$

2. **References :** [LL73].
3. **Assumptions :**
   – Request on deadline tasks.
   – Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   – If task are not harmonic, then $B = n(2^{\frac{1}{n}} - 1)$ and the test is sufficient but not necessary condition.
   – If task are harmonic, then $B = 1$ and the test is a sufficient and necessary test.

5.2 C2 test : preemptive Earliest Deadline First or Least Laxity First

1. **How to compute :**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

2. **References :** [LL73, LM80, CDKM00].
3. **Assumptions :**
   – Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   – Necessary and sufficient condition if $\forall i : D_i = P_i$.
   – If $\exists i : D_i < P_i$, then $\sum_{i=1}^{n} \frac{C_i}{D_i} \leq 1$ is a sufficient condition only and $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$ is a necessary condition only.

5.3 C3 test : non preemptive Rate Monotonic

1. **How to compute :**

$$\forall\ i, 1 \leq i \leq n : \sum_{i=1}^{n} \frac{C_i}{P_i} + \max_{i<i\leq n} \left(\frac{B_i}{P_i} \right) \leq n(2^{\frac{1}{n}} - 1)$$

2. **References :** [Car96].
3. **Assumptions :**
   - Request on deadline tasks.
   - Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   - Condition suffisante mais non ncessaire.
   - On suppose que les tches sont ordonnes de faon dcroissante selon leur priorit : la tche $i-1$ est donc moins prioritaire que la tche i.

5.4 C4 test : preemptive and non preemptive Rate Monotonic

1. **How to compute :**

$$\forall\ i, 1 \leq i \leq n : \sum_{k=1}^{i-1} \frac{C_k}{P_k} + \frac{C_i + B_i}{P_i} \leq i(2^{\frac{1}{i}} - 1)$$

2. **References :** [SRL90, Car96].
3. **Assumptions :**
   - Pour Rate Monotonic non premptif : tches indpendantes, chances sur requte, concrtes et synchrones.
   - Pour Rate Monotonic premptif : tches dpendantes, chances sur requte, concrtes et synchrones.
4. **Comments :**
   - Condition suffisante mais non ncessaire.
   - On suppose que les tches sont ordonnes de faon dcroissante selon leur priorit : la tche $i-1$ est donc moins prioritaire que la tche i.
   - Ce test peut tre employ dans deux cas de figures :
     (a) Soit pour tester un jeu de tches ordonnances avec un algorithme Rate Monotonic premptif quand les tches accdent des ressources partages. On suppose alors que B_i est calcul en fonction du protocole d'accs aux ressources partages.
     (b) Soit pour tester un jeu de tches indpendantes avec un algorithme Rate Monotonic non premptif. Dans ce cas, on suppose que B_i est le temps d'attente de la tche i provoqu par les tches de plus faibles priorits.

5.5 C5 test : preemptive Deadline Monotonic

1. **How to compute :**

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1)$$

2. **References :** [LL73].
3. **Assumptions :**
   - Tasks with $\forall i : D_i \leq P_i$.
   - Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   - Condition suffisante mais non ncessaire.

5.6 C6 test : non preemptive Earliest Deadline First

1. **How to compute :**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

   et

$$\forall\ 1 < i \leq n : C_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{P_j} \right\rfloor \cdot C_j \leq L$$

   avec $P_1 < L < P_i$
2. **References :** [JDM91].
3. **Assumptions :**
   - Request on deadline tasks.
   - Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   - Necessary and sufficient condition.
   - On suppose que les tches sont ordonnes de faon dcroissante selon leur priorit : la tche $i-1$ est donc moins prioritaire que la tche i.

5.7 C7 test : any preemptive fixed priority scheduler

1. **How to compute :**

$$\forall\ 1 \leq i \leq n : min_{0 \leq t \leq D_i} \left(\sum_{j=1}^{i} \frac{C_j}{t} \cdot \left\lceil \frac{t}{P_j} \right\rceil \right) \leq 1$$

2. **References :** [LSD89].
3. **Assumptions :**
   - Algorithme priorit fixe, quelque soit l'algorithme d'affectation des priorits.
   - Request on deadline tasks.
   - Independent periodic tasks. Concrete and synchronous task set.
4. **Comments :**
   - Condition ncessaire et suffisante.
   - On suppose que les tches sont ordonnes de faon dcroissante selon leur priorit : la tche $i-1$ est donc moins prioritaire que la tche i.

5.8 C8 test : preemptive Earliest Deadline First

1. **How to compute :**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

et

$$\forall t \geq 0 : h(t) \leq t$$

avec $h(t)$, la demande processeur tel que :

$$h(t) = \sum_{D_i \leq t} \left(1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor \right) \cdot C_i$$

2. **References :** [BHR90, GRS96].
3. **Assumptions :**
   - Tasks with $\forall i : D_i \geq P_i$.
   - Independent periodic tasks. Non concrete and synchronous task set.
4. **Comments :**
   - Necessary and sufficient condition.

6 Tests based on worst case response times

6.1 R1 test : any preemptive fixed priority scheduler

1. **How to compute :**

$$r_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{r_i}{P_j} \right\rceil \cdot C_j$$

2. **References :** [JP86].
3. **Assumptions :**
   - Algorithme premptif priorit fixe, quelque soit l'algorithme d'affectation des priorits.
   - Tches chances sur requtes.
   - Tches indpendantes, concrtes et synchrones.
4. **Comments :**
   - $hp(i)$ est l'ensemble des tches de plus forte priorit que i.
   - Necessary and sufficient condition.

6.2 R2 test : any preemptive fixed priority scheduler

1. **How to compute :**

$$r_i = \max_{q=0,1,2,\ldots} (J_i + B_i + w_i(q) - q \cdot P_i)$$

avec

$$w_i(q) = (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + w_i(q)}{P_j} \right\rceil \cdot C_j$$

et

$$\forall q : w_i(q) \geq (q+1) \cdot P_i$$

2. **References :** [JP86, ABRT93, TC94].
3. **Assumptions :**
   - Algorithme premptif priorit fixe, quelque soit l'algorithme d'affectation des priorits.
   - Tches chances quelconques, concrtes et synchrones.
   - Tches dpendantes (ressources partages et/ou contraintes de prcdence exprime avec une gigue).
4. **Comments :**
   - $hp(i)$ est l'ensemble des tches de plus forte priorit que i.
   - Condition ncessaire et suffisante si la gigue et le temps de blocage sur ressources partages ne sont pas employs. Condition suffisante seulement dans le cas contraire.

6.3 R3 test : preemptive Earliest Deadline First premptif

1. **How to compute :**

$$r_i = max_{a \in S}(L_i(a) - a)$$

avec :

$$L_i(a) = W(a, L_i(a)) + \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) \cdot C_i$$

$$S = \bigcup_{j=1}^{n} \left(k \cdot T_j + D_j - D_i, 0 \leq k \leq \left\lfloor \frac{min(\lambda, L_i)}{T_j} \right\rfloor\right)$$

$$\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot C_j$$

2. **References :** [Spu96, GRS96].
3. **Assumptions :**
   - Tches chances quelconques.
   - Tches non concrtes.
4. **Comments :**
   - Necessary and sufficient condition.

6.4 R4 test : any non preemptive fixed priority scheduler

1. **How to compute :**

$$r_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{P_j} \right\rceil \cdot C_j + max(C_k, \forall k \in lp(i))$$

2. **References :** [JP86, GRS96].
3. **Assumptions :**
   - Algorithme premptif priorit fixe, quelque soit l'algorithme d'affectation des priorits.
   - Tches chances sur requtes.
   - Tches indpendantes, concrtes et synchrones.
4. **Comments :**
   - $hp(i)$ est l'ensemble des tches de plus forte priorit que i.
   - $lp(i)$, l'ensemble des tches de plus faible priorit que i.
   - Sufficient condition only.

6.5 R5 test : non preemptive Earliest Deadline First

1. **How to compute :**

$$r_i = max_{a \in S}(C_i, L_i(a) - a)$$

avec :

$$L_i(a) = max_{D_j > a + D_i}(C_j) + \sum_{j \neq i, D_j \leq a + D_i} min\left(1 + \left\lfloor \frac{L_i(a)}{T_j} \right\rfloor, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right) \cdot C_j + \left\lfloor \frac{a}{T_i} \right\rfloor \cdot C_i$$

$$S = \bigcup_{j=1}^{n} \left(k \cdot T_j + D_j - D_i, 0 \leq k \leq \left\lfloor \frac{min(\lambda, L_i)}{T_j} \right\rfloor\right)$$

$$\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot C_j$$

2. **References :** [Spu96, GRS96].
3. **Assumptions :**
   - Task with any type of deadlines.
   - Non concrete task set.
4. **Comments :**
   - Necessary and sufficient condition.

7 Computing of task blocking time on shared resources

Actually, this last section does not contain feasibility tests but equations that could be used in some feasibility tests. In the sequel, we list methods that Cheddar apply to compute task blocking time on shared resources. This delay is sometimes including in worst case response time computation.

7.1 P1 test : Priority Inversion Protocol

1. **How to compute :**
   $B_i =$ is the sum of the duraion of all critical sections of the tasks with a priority level lower than i's priority.
2. **References :** [SRL90].
3. **Comments :**
   - Compute task blocking time of task i according to PIP.
   - Only one shared resource between the task,otherwise this protocol leads to deadlock.

7.2 P2 test : Priority Ceiling Protocol

1. **How to compute :**
   $B_i =$ is the longest critical section of the tasks sharing the same set of resources.
2. **References :** [SRL90].
3. **Comments :**
   - Compute task blocking time of task i according to any version of PCP such as OCPP or ICPP.
   - Several shared resources without deadlock.

7.3 P3 test : Stack Resource Protocol

1. **How to compute :**
   $B_i =$ is the longest critical section of the tasks with
   (a) With a period which is greater than the period of task i.
   (b) With a preemption level which is greater than preemption level of task i.
2. **References :** [Bak91, GN01].
3. **Comments :**
   - Compute task blocking time of task i according to SRP.
   - Several shared resources without deadlock.

Références

[ABRT93] A. N. Audsley, A. Burns, M. Richardson, and K. Tindell. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, pages 284–292, 1993.

[Bak91] T.P. Baker. Stack-based scheduling for realtime processes. *Journal of Real Time systems*, 3(1) :67–99, March 1991.

[BHR90] S.K. Baruah, R. R. Howell, and L. E. Rosier. Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real Time Tasks on one Processor. *Real Time Systems journal*, 2 :301–324, 1990.

[Car96] F. V. Carvahlo. Sur l'Intégration de Mécanismes d'Ordonnancement et de Communication dans la sous-Couche MAC de Réseaux Locaux Temps réel . Thèse de l'Université de Toulouse 3, 1996.

[CDKM00] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Ordonnancement temps réel.* Hermès, 2000.

[GN01] P. Gai and M. Di Natale. Minimizing memory utilization of real-time task sets in single and multi-processor systems-on-a-chip . volume 3, pages 67–99. In Proceedings of the 22nd IEEE Real-Time Systems Symposium, March 2001.

[GRS96] L. George, N. Rivierre, and M. Spuri. Preemptive and Non-Preemptive Real-time Uni-processor Scheduling. INRIA Technical report number 2966, 1996.

[JDM91] K. Jeffay, D.Stanat, and C. Martel. On non preemptive Scheduling of periodic and Sporadic Tasks. in Proc. Of the IEEE Real Time Systems Symposium (RTSS'91), San Antonio, Texas,, December 1991.

[JP86] M. Joseph and P. Pandya. Finding Response Time in a Real-Time System. *Computer Journal*, 29(5) :390–395, 1986.

[KRP+94] M. H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour. *A Practitioner's Handbook for Real Time Analysis.* Kluwer Academic Publishers, 1994.

[LL73] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnment. *Journal of the Association for Computing Machinery*, 20(1) :46–61, January 1973.

[LM80] J.Y.T Leung and M.L. Merril. A note on preemptive scheduling of periodic real time tasks. *Information processing Letters*, 3(11) :115–118, 1980.

[LSD89] J. Lehoczky, L. Sha, and Y. Ding. The Rate Monotonic Scheduling Algorithm : Exact Characterization and Average Case Behaviour. pages 166–171. in Proc. Of the IEEE Real Time System symposium (RTSS'89), December 1989.

[RRC02] P. Richard, M. Richard, and F. Cottet. *Analyse holistique des systèmes temps réel distribués : principes et algorithmes.* Ordonnancement dans les systèmes distribués, Hermès, 2002.

[Spu96] Marco Spuri. Analysis of deadline scheduled real-time systems. Technical Report RR-2772, 1996.

[SRL90] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority Inheritance Protocols : An Approach to real-time Synchronization. *IEEE Transactions on computers*, 39(9) :1175–1185, 1990.

[TC94] K. W. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2-3) :117–134, April 1994.