Le patch RT_Preempt et RTAI

Eurogiciel

Agence de Rennes
22 rue Rigourdière - 35510 Cesson Sévigné

26 janvier 2011

Objectifs

- Préemptibilité de la majeure partie du code du noyau
- Possibilité de préemption des sections critiques
- Possibilité de préemption des gestionnaires d'interruption
- Réduction des temps de latence
- Protection contre les inversions de priorité
- Conservation de l'API Posix standard

Performances

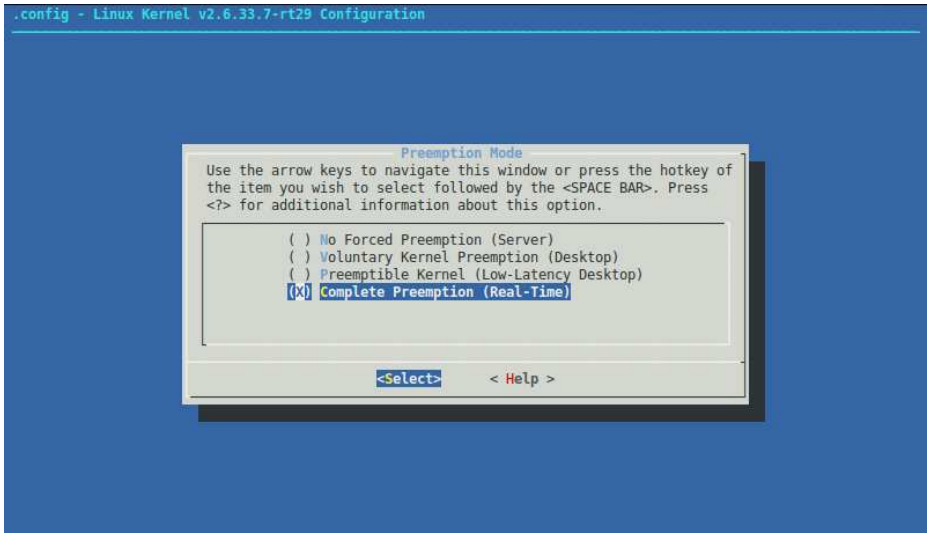
- Temps de latence moyen de l'ordre de 20 à 30 microsecondes
- Temps de latence maximum de l'ordre de 100 à 500 microsecondes (dépendant de la configuration matérielle et logicielle)

Configuration du noyau (1/2)

- Ajout d'une option `CONFIG_PREEMPT_RT`
- Apport d'un mode de préemption supplémentaire

Configuration du noyau (2/2)

- Sélection de l'option "Complete Preemption (Real-Time)"
- Sélection de l'option "High Resolution Timer Support" (`CONFIG_HIGH_RES_TIMERS`)



Récupération du patch

- Placement dans le répertoire des sources des noyaux
  `cd /usr/src`
- Téléchargement des sources
  `wget http://www.kernel.org/pub/linux/kernel/projects \`
  `/rt/patch-2.6.33.7-rt29.bz2`

Application du patch

- Exécution en mode superutilisateur (root)
- Déplacement dans le répertoire du noyau concerné
  `cd linux-2.6.33.7`
- Test de compatibilité
  `bzcat ../patch-2.6.33.7-rt29.bz2 | patch -p1 --dry-run`
- Modification des sources du noyau
  `bzcat ../patch-2.6.33.7-rt29.bz2 | patch -p1`
- Recompilation du noyau

Processeur non chargé

Noyau	Latence moyenne (μs)	Latence maximale (μs)
Standard	7	553
Patché	14	31

Processeur chargé

Noyau	Latence moyenne (μs)	Latence maximale (μs)
Standard	23	865
Patché	4	48

Conditions d'expérimentation

- Test indicatif et non généralisable
- Réalisation sur un matériel unique
- Jeu de test réduit

Constatations

- Réduction des temps de latence
- Gains importants en situation de charge du processeur
- Limitation de la variance entre les résulats moyens et maximaux

2 Introduction à RTAI
- Présentation
- Mise en œuvre
- Évaluation

Objectifs

- Apport de possibilités pour le temps réel dur
- Préemption totale du noyau standard
- Réduction de la latence
- Très faible gigue
- Possibilité de développement en espaces utilisateur et noyau

Principe d'intégration

- Implantation d'un co-noyau
- Considération du noyau standard comme une tâche de bas niveau
- Mise à disposition d'une API spécifique
- Support d'applications au standard Posix

Récupération du patch

- Placement dans le répertoire des sources des noyaux
  `cd /usr/src`
- Téléchargement des sources
  `wget https://www.rtai.org/RTAI/rtai-3.8.tar.bz2`

Application du patch

- Décompression de sources
  `tar -xvjf rtai-3.8.tar.bz2`
- Déplacement dans le répertoire du noyau concerné
  `cd linux-2.6.23`
- Modification des sources du noyau
  ```
  patch -p1 < /usr/src/rtai-3.8/base/arch/i386/ \
  patches/hal-linux-2.6.23-i386-1.12-03.patch
  ```
- Recompilation du noyau

Test de latence

- Passage en mode `root`
- Placement dans le répertoire des utilitaires de test
  `cd /usr/realtime/testsuite/kern/latency/`
- Lancement du test
  `./run`
- Résultats de l'ordre de 10 à 15 microsecondes (selon configuration)

3 Conclusion

Bilan du patch RT_Preempt

- Amélioration globale des temps de latencet
- Contexte d'utilisation plutôt limité au temps réel mou
- Solution viable pour la plupart des applications
- Exécution des programmes compilés pour la version standard du noyau

Bilan de RTAI

- Amélioration de la latence et de la gigue
- Contexte d'utilisation adapté au temps réel dur
- Développement via une API spécifique

Questions

?