# Systèmes et supports d'exécution pour le calcul parallèle et distribué

Université de Bordeaux 1 — Master 2 SDRP

Examen 2006–2007

Alexandre DENIS

19 janvier 2007

Durée : 1 h 30. Documents autorisés.
Les exercices sont indépendants les uns des autres. Le barème est donné à titre indicatif. Les parties de code demandées doivent être rédigées en langage C.

## 1 Multi-threading (7 points)

1. Qu'est-ce qu'un ordonnanceur de *threads* ? Du point de vue de l'ordonnanceur, qu'est-ce qu'un *thread* ?
2. Quels sont les types d'ordonnanceurs de *threads* existants ? Rappeller leurs propriétés, en justifiant.
3. Que sont les *spinlock* et *mutex* ? Quelles sont leurs différences ? Dans quel cas utilise-t-on l'un ou l'autre ?
4. Dans quel cas utilise-t-on ou non un *spinlock* pour implémenter un *mutex* ? Pourquoi ?
5. Proposer une implémentation de sémaphore (opérations `sem_P` et `sem_V`) à partir de *spinlocks*.

## 2 Réseaux haute performance (8 points)

1. Qu'est-ce que le RDMA ? Expliquez, en vous aidant de figures si besoin, les performances relatives d'une écriture distante et d'une lecture distante en RDMA.
2. Que sont les *Memory Regions* (MR) d'*Infiniband* ? À quoi servent-elles ? Où et comment ce mécanisme est-il mis en œuvre dans le réseau *Myrinet* avec MX ?
3. Pour envoyer les données par RDMA, il est possible de copier les données en MR, puis d'effectuer un seul envoi sur le réseau ; une autre technique consiste à fragmenter les données en morceaux de petite taille, copiés en MR et envoyés individuellement, sous forme de *pipeline*.

   *Note : dans cette question, on supposera que l'émetteur effectue un double buffering (envoie sur le réseau pendant qu'il copie le fragment suivant), et on négligera les conflits sur le bus mémoire liés à ces opérations concurrentes. Le récepteur est supposé avoir une fenêtre de réception assez large pour recevoir tous les fragments.*

   (a) Représenter ces deux solutions sur un diagramme temporel.

   (b) Si on appelle $D$ le débit nominal de la carte *Infiniband*, $\lambda$ sa latence, $D_{copie}$ le débit d'une copie en mémoire, exprimer $T(F)$ le temps d'envoi de la méthode qui fragmente les messages, en fonction de la taille des fragments notée $F$ et de la taille du message notée $L$. Qu'en déduisez-vous sur la valeur optimale de $F$ ? Est-ce réaliste ?

   (c) En réalité, le coût logiciel pour poster une requête d'envoi par la carte *Infiniband* n'est pas nul, c'est-à-dire qu'envoyer deux paquets de taille $L$ est plus long qu'envoyer un paquet de taille $2 \times L$. Notons $\varepsilon$ cette différence de temps, correspondant à l'intervalle de temps entre deux paquets sur le réseau. Ajouter la prise en compte de $\varepsilon$ dans la formule exprimant le temps d'envoi $T(F)$.

   (d) Exprimer la taille idéale de fragment $F_0$, telle que $T(F_0)$ soit le minimum de $T(F)$, en fonction de $L$ la longueur du message.

   (e) *Bonus : application numérique.* Sachant que sur les machines *Infiniband* du Cremi, $\lambda = 3.2\mu s$, $D = 1.4$ Go/s, $D_{copie} = 2.1$ Go/s, et $\varepsilon = 0.3\mu s$, quelle est la taille idéale de fragment pour un message de 1 Mo ? Quel est alors le débit obtenu ?
4. On prend maintenant en compte le conflit sur le bus mémoire.

   (a) Expliquer le problème.

   (b) Quel débit effectif attendre de la part du réseau ($D$) et pour les copies ($D_{copie}$) ? Faire une application numérique sachant que le bus mémoire est à 4.2 Go/s sur les machines considérées.

   (c) Quel est alors le débit asymptotique ($L \to \infty$) obtenu par la méthode *pipeline* décrite ci-dessus, en supposant $F$ fixé ?
5. En plus du mode RDMA, les cartes *Infiniband* sont également capables d'échanger des messages en mode *send/recv*. En mode *send/recv*, si le récepteur n'est pas prêt à recevoir, le paquet est perdu, un *timeout* expire côté émetteur, et le paquet est réémis ; c'est un évènement RNR (*Receiver Not Ready*).

   (a) Quelles sont les différences entre ce mode et le RDMA ? Pourquoi le paquet est-il perdu en cas de RNR ?

   (b) Quels sont les inconvénients d'utiliser un *timeout* ?

   (c) Proposer un mécanisme permettant d'éviter les RNR, ainsi qu'une évaluation de son coût.

   (d) Y a-t-il risque de RNR en RDMA ? Justifiez.

   (e) Si le récepteur n'est pas prêt, où sont les données si l'envoi est effectué en RDMA ? En mode *send/recv* ? Quel mode semble le mieux adapté à ce cas ? Justifiez.

## 3 Systèmes à grande échelle (5 points)

1. Quelles sont les différences fondamentales entre *pair-à-pair* et client-serveur ? À quels cas chaque approche est-elle adaptée ?
2. En *pair-à-pair*, qu'appelle-t-on une DHT ? Que contient-elle ? Expliquer son fonctionnement et comment elle est utilisée lors du routage.
3. Expliquer, en vous appuyant sur une figure, ce que sont les souches (*stubs*) et squelettes (*skeletons*), en vous basant éventuellement sur l'exemple de `gSoap`. D'où sont-ils issus ? Quel est leur rôle ? Que devraient faire les programmeurs d'applications sinon ?