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

Université de Bordeaux 1 — Master 2 SDRP

Examen 2005–2006

Alexandre DENIS

20 janvier 2006

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 Questions de cours (6 points)

1. Qu'appelle-t-on la *portabilité* ? Quelles sont les différentes méthodes pour l'assurer ? Dans quels cas est-ce plus ou moins important ?
2. Qu'appelle-t-on l'*hétérogénéité* ? À quels niveaux se situe-t-elle ? En quoi est-elle un problème, et quelles sont les solutions ?
3. Que sont les machines virtuelles ? Dans quel but sont-elle conçues ?
4. Quelle est la différence entre un système parallèle et un système réparti ? Quelles différences sur leurs propriétés en découlent ?

2 Grappes (5 points)

1. Qu'appelle-t-on une grappe de PC ?
2. Quels sont les avantages et inconvénients relatifs des grappes et des supercalculateurs ?
3. Le protocole de MX sur le réseau Myrinet dispose de trois modes d'envois selon la taille du message : *rendez-vous*, PIO, DMA.
   (a) Expliquez à quoi correspondent ces modes.
   (b) Quel est l'intérêt d'utiliser plusieurs modes ?
   (c) Que risquerait-il de se passer sans mode *rendez-vous* ?
   (d) À quelle taile de paquet (petits, moyens, gros) correspond chacun des trois modes ? Justifiez.
   (e) Si on appelle D le débit nominal de la carte Myrinet, λ sa latence, D_{PIO} le débit du bus PCI en mode PIO, et t_{DMA} le temps d'initialisation d'une requête DMA, écrivez (sans la résoudre) l'équation qui permet de déterminer le seuil idéal de basculement entre DMA et PIO.

3 Multi-threading (5 points)

Soit l'extrait de code C suivant :

```
double s = 0;
for(i = 0; i < size ; i++)
{
  s += v[i];
}
```

1. Que calcule ce programme ?
2. Quelles sont les techniques de parallélisation qui lui sont applicables sur une machine SMP ? Justifiez.
3. On s'intéresse maintenant à la version Open MP du même programme :

```
double s = 0;
#pragma omp parallel for
#pragma omp private(i), shared(s, v)
for(i = 0; i < size ; i++)
{
  s += v[i];
}
```

   Que signifient les lignes ajoutées ? À qui s'adressent-elles ?
4. Transformez ce programme en code `pthread` de la même façon que le ferait automatiquement un compilateur Open MP, de façon à ce qu'il s'exécute sur `NPROCS` processeurs. On pourra supposer que `size` est divisible par `NPROCS`. Considérer les deux cas suivants :
   (a) un mutex coûte plus cher qu'une addition de `double;`
   (b) un mutex coûte moins cher qu'une addition de `double`.

4 Pair-à-pair (4 points)

1. Qu'appelle-t-on *pair-à-pair* ? Quelles sont les motivations pour l'utiliser ?
2. Comment sont établies les connexions ? Y en-t-il une établie de chaque pair à chaque autre pair ? Pourquoi ?
3. Comment les pairs qui n'ont pas de connexion directe entre eux échangent-ils des messages ?
4. Quels types d'applications vous semblent appropriés pour une approche pair-à-pair (à part le partage de fichier, bien entendu) ? Justifiez.
5. On souhaite écrire une application de *stockage de données* (à la façon NFS, données modifiables à distance) sur un réseau pair-à-pair. Pour gérer la volatilité des pairs sans perdre de données, les données sont répliquées sur plusieurs pairs. Proposer les grandes lignes d'un mécanisme assurant la cohérence des copies, en vous inspirant des mécanismes de cohérence de cache.