TP 7 : Premier contact avec *Madeleine*

Alexandre DENIS
denis@labri.fr

Systèmes et supports d'exécution — 2004–2005

L'objectif de ce TP est de découvrir la bibliothèque de communication *Madeleine* et de préparer au prochain TP — « *multiplexage sur Madeleine* » — qui sera **noté**.

1 La bibliothèque *Madeleine*

1.1 Principe

Madeleine est une bibliothèque de communication portable et générique, capable d'utiliser efficacement des réseaux haute performance tels que *Myrinet* ou SCI. Une des originalité de *Madeleine* est que l'interface est construite à l'aide de contraintes exprimées au niveau *sémantique* (« *cheaper* », « *express* », etc.) et non de manière impérative (« *en-tête* », « *corps de message* », etc.) comme la plupart des autres bibliothèques de communication.

Le modèle proposé repose sur la notion de *canaux* qui, de façon semblable aux *communicateurs* MPI, sont composés d'un ensemble de nœuds et permettent les communications de chacun des nœuds vers chaque autre nœud du canal. À l'intérieur d'un canal, les nœuds sont identifiés par un numéro logique ; la numérotation commence à 0. Dans ce TP, nous n'utiliserons qu'un seul canal.

Madeleine est développée dans l'équipe de recherche *Runtime* du *LaBRI*. Vous trouverez des informations complémentaires à l'adresse :
`http://dept-info.labri.fr/~aumage/Runtime/logiciels.html`

1.2 Interface de programmation

L'échange de messages est effectué à l'aide d'une interface de construction de messages *incrémentale* :

`mad_begin_packing(canal, destinataire)` débute un envoi de message sur le canal et vers le destinataire donné. La fonction renvoie un objet *connexion*.

`mad_pack(connexion, buffer, size, mode_envoi, mode_recéption)` ajoute un paquet à un message commencé par un `mad_begin_packing`.

`mad_end_packing(connexion)` termine un message.

Des contraintes sémantiques sont données pour préciser *quand* Madeleine accède aux buffers, en donnant un mode d'émission et un mode de réception pour chaque paquet. Le mode d'émission détermine les garanties données à l'émetteur ; le mode de réception détermine les garanties côté récepteur.

mode d'émission	*sémantique*
`mad_send_SAFER`	la valeur envoyée est celle au moment du `mad_pack`
`mad_send_LATER`	la valeur envoyée est celle au moment du `mad_end_packing`
`mad_send_CHEAPER`	aucune garantie avant le `mad_end_packing`

mode de réception	*sémantique*
`mad_receive_EXPRESS`	les données sont disponibles immédiatement après le `mad_unpack`
`mad_receive_CHEAPER`	aucune garantie avant le `mad_end_unpacking`

Il va de soit que les modes « *cheaper* » pour lesquels il est interdit de toucher aux données entre le `begin_packing` et le `end_packing` autorisent de meilleures performances, car *Madeleine* peut alors réordonnancer ou agréger les paquets, mais ces modes ne conviennt pas à tous les cas.

Chaque primitive d'émission à son symétrique pour la réception. On utilisra ainsi :

`mad_begin_unpacking(canal)` pour attendre un nouveau message. *Note* : cette fonction est bloquante et ne permet pas de choisir la source du message — on est obligé de traiter le message qui arrive, même si ce n'est pas celui qui était attendu.

`mad_unpack(connexion, buffer, size, mode_envoi, mode_recéption)` reçoit un paquet

`mad_end_unpacking(connexion)` termine la réception du message

Attention, il est primordial que les `mad_pack` et `mad_unpack` soient symétrique : même taille, même modes. En effet, *Madeleine* n'envoie pas d'en-tête[1] sur le réseau donc ne peut pas retrouver les frontières de paquets sans votre aide.

1.3 Exemple

Par exemple, pour envoyer des données stockées dans `buffer` de taille `size` :

```
p_mad_connection_t connection = mad_begin_packing(channel, dest);
mad_pack(connection, buffer, size, mad_send_CHEAPER, mad_receive_CHEAPER);
mad_end_packing(connection);
```

Et côté récepteur :

```
p_mad_connection_t connection = mad_begin_unpacking(channel);
mad_unpack(connection, buffer, size, mad_send_CHEAPER, mad_receive_CHEAPER);
mad_end_unpacking(connection);
```

1.4 Initialisation

Reprendre tel-quel le code suivant pour initialiser un processus *Madeleine* :

```
p_mad_madeleine_t madeleine = NULL;
p_mad_session_t   session   = NULL;
p_mad_channel_t   channel   = NULL;
int num_nodes, my_rank;
```

[1] À votre avis, pourquoi ?

```
common_init(&argc, argv, NULL);

madeleine = mad_get_madeleine();
session   = madeleine->session;
my_rank   = session->process_rank;
num_nodes = tbx_slist_get_length(madeleine->dir->process_slist);
channel   = mad_get_channel(madeleine, "pm2");
```

Ne pas oublier de synchroniser la terminaison par :

```
common_exit(NULL);
```

2 Travail à faire

2.1 Envoi simple

Ecrire un programme émetteur et un programme récepteur s'échangeant un message de 10 octets (taille fixe et connue par les deux parties).

2.2 Ping-pong

Étendre le programme précédent pour réaliser un ping-pong, de façon à pouvoir mesurer les performances.

2.3 Benchmark

Modifier le programme de façon à envoyer des messages de taille croissante (les puissances de 2, de 4 octets à 4 Mo par exemple), et mesurer le temps de transfert pour chacune des tailles. Tracer ensuite la courbe de débit en fonction de la taille des messages à l'aide de `gnuplot`.

2.4 Comparaison

Finalement, *Madeleine* et GM offre le même service : la possibilité d'échanger des messages sur Myrinet.

- Comparer la facilité de programmation par rapport à une programmation directe de GM !
- Comparer les performances obtenues par rapport à la programmation directe de GM.
- Selon vous, est-il plus « rentable » d'utiliser *Madeleine* ou GM ? À part la facilité de développement, quel est l'autre avantage d'utiliser *Madeleine* ?

2.5 Taille arbitraire

Nous allons maintenant nous rapprocher de l'utilisation de *Madeleine* dans les conditions *réelles*.

- Modifier votre programme de façon à transmettre en en-tête la taille des données — seul l'émetteur connaît la taille des messages, le récepteur l'extrait des en-têtes.
- Comparer les performances avec la version où le récepteur connaît à l'avance la taille des données. Conclusion ?

2.6 Travail facultatif

Pour les plus téméraires :

- Implémenter un protocole à *rendez-vous* au-dessus de *Madeleine*.
- Le transformer en protocole multi-méthode en fonction de la taille des données.
- Déterminer de façon expérimentale et théorique le seuil (*on prendra* $D = 12$ *Mo/s pour Ethernet,* $D = 240$ *Mo/s pour Myrinet*).