Paradigme de passage de message

- messages = paquets de données entre sous-programmes
- Le système de passage de message doit avoir les informations suivantes :
  - processeur qui envoie
  - localisation de la source
  - type de données
  - taille des données
  - processeur qui reçoit
  - localisation de la destination
- Accès : un sous programme doit être connecté au système de passage de message

Communication point à point

- La forme la plus simple du passage de message
- Un processus (ou tâche) envoie un message à un autre
- Différents types de communication point à point
  - Envoi synchrone : fournit de l'information sur le fait que le message ait été effectivement reçu
  - Envoi asynchrone : on sait juste que le message a été envoyé

Opérations (non)bloquantes

- bloquante :
  - relative au moment où l'opération se termine
  - on ne reprend la main qu'au retour de l'opération bloquante
- non bloquante :
  - on lance l'opération, puis le programme continue. Après un certain temps le programme peut tester la fin de l'opération ou l'attendre

Exemple

- Barrière de synchronisation : synchronisation des processus
- Broadcast : communication de un vers plusieurs
- Réduction : combiner les données issues de plusieurs processeurs (machines) pour produire un seul résultat

Paralléliser une application

- On a une application :
  - regarder la physique
  - examiner les structures de boucles
  - si algèbre linéaire alors utiliser les versions // des algorithmes
- ne pas ré-inventer la roue
- schémas de programmation usuel :
  - maître/esclave
  - SPMD
  - fonctionnel : chaque programme traite une partie de l'application

Le SPMD

- Single Program Multiple Data
- le même programme tourne partout
- Restriction du modèle de passage de messages

Considération de programmation //

- Granularité des tâches
- Nombre de messages
- Latence des messages
- Machines hétérogènes de puissances variables et de tailles mémoires variables
- Fluctuation des charges réseau et machine

PVM est un paquetage de soft

- le supercalculateur // du pauvre
- permet le méta-computing en reliant plusieurs supercalculateurs
- outil éducatif

Le système PVM (1)

- le démon PVM (pvmd) :
  - un pour chaque machine physique de la machine virtuelle
  - un routeur de messages
  - des points d'entrée pour les messages, les tâches y font leurs requêtes
  - point de contact entre machines
  - authentification
  - création de processus/tâche
  - collecte les sorties imprimées par les processus
  - détection des erreurs de réseau et de processus
- une librairie d'inteface (libpvm.a) :
  - à lier avec tous les composants de l'application //
  - fonction de composition, d'envoi et réception de messages

Le système PVM (2)

- la console PVM (pvm) :
  - contrôle interactif de la machine virtuelle
  - une sorte de shell pour la machine virtuelle
- XPVM l'interface graphique de PVM :
  - même fonction que la console
  - moniteur temps réel graphique avec :
    * vue de la configuration de la machine virtuelle et de son activité
    * diagramme espace/temps des états des tâches
    * affichage de l'utilisation des machines physiques
    * appel du debugger, visualisation des appels de la librairie pour chaque tâche
  - génération de fichier de trace
  - permet l'analyse post-mortem d'une exécution
  - construit au dessus de PVM en utilisant :
    * la librairie de groupe
    * le système de trace

Programmer en PVM (1)

- Un environnement de passage de message simple :
  - machine, tâche, message
  - pas de topologie par défaut
  - la machine virtuelle peut se composer de n'importe quelle machine
- contrôle de processus :
  - les tâches peuvent être tuées/spawnées n'importe où dans la machine virtuelle
- communication :
  - une tâche peut communiquer avec n'importe quelles autres
  - convertion de données prises en charge par PVM

Programmer en PVM (2)

- groupe de processus dynamique :
  - une tâche peut se joindre ou quitter un ou plusieurs groupes à n'importe quel moment
- tolérance aux fautes :
  - une tâche peut demander l'état (perte/gain) des ressources
- langages : C, C++ et Fortran