Parallel Virtual Machine (PVM)

1. Introduction (see Geist, et al., 1994)

- The challenge of heterogeneous distributed computing:
  - architecture
  - data format
  - operating system
  - computational speed
  - machine load
  - network load

- The advantages of heterogeneous distributed computing:
  - Use existing hardware.
  - Use most appropriate hardware.
  - Virtual computer can be easily upgraded.
  - Program development on familiar platforms.
  - Exploit stability of workstations.
  - Facilitate collaborative work.

- Some principles underlying PVM:
  - User-configured host pool.
  - Translucent access to hardware.
  - Process-based computation.
  - Explicit message-passing model.
  - Heterogeneity supported.
  - Multiprocessor support.

- There are two major components in the system:
  (a) A daemon which runs on every machine in the virtual machine.
  (b) A library of routines (for message passing, spawning and coordinating tasks, modifying the virtual machine, etc.).
2. Programming in PVM

Process control:
- `mytid` returns the `tid` of the process
- `exit` tells the local pvm that this process is leaving PVM
- `spawn` start new tasks
- `kill` terminates a given task

Information:
- `parent` get `tid` of my parent
- `tidtohost` get host id where specified task is running
- `config` get info about virtual machine configuration
- `tasks` get info about current tasks

Dynamic configuration:
- `addhosts` add hosts to virtual machine
- `delhosts` delete hosts from virtual machine
- `mstat` get status of specified host

Signaling:
- `sendsig` send signal to another task
- `notify` request notification of PVM event

Options:
- `setopt` set various options
- `getopt` returns current option settings

Message Passing:
- `initmsg` clears send buffer and specifies message encoding
- `mkbuf` create new message buffer
- `pkbyte` pack an array of bytes into active send buffer
- `send` send data in active message buffer
- `mcast` Multicast to a set of tasks
- `pseq` packs and sends data
- `recv` blocking receive
- `recv` nonblocking receive
- `probe` check for message arrival
- `trecv` blocking receive with timeout
- `precv` combines blocking receive and unpacking
- `upkbyte` unpack array of bytes

Dynamic process groups:
- `joingroup` enroll calling process in named group
- `lvgroup` unenrolls calling process from group
- `gettid` get `tid` given group and instance
- `getinst` get instance number of a task
- `gsze` get size of group
- `barrier` block calling process until all processes in a group have called it
- `bcast` broadcast message to all members of group
- `reduce` reduced operation over a group