Last time

- Coordination for parallel programs
  - synchronization
  - load balancing
- Control vs. data parallelism
- Metrics
  - Speedup - vs. best known serial algorithm
  - Scaled speedup
  - Amdahl's law
  - Maximize computation to communication ratio
- Writing parallel programs
  - compiler converts old serial code
  - serial language plus communication library
  - new programming language
  - hybrid - old language with new constructs

PVM

- Provide a simple, free, portable parallel environment
- Run on everything
  - Parallel Hardware: SMP, MPPs, Vector Machines
  - Network of Workstations: ATM, Ethernet,
    - UNIX machines and PCs running Win32 API
  - Works on a heterogenous collection of machines
    - handles type conversion as needed
- Provides two things
  - message passing library
    - point-to-point messages
    - synchronization: barriers, reductions
  - OS support
    - process creation (pvm_spawn)
**PVM Environment (UNIX)**

- One PVMD per machine
  - all processes communicate through pvmd (by default)
- Any number of application processes per node

**PVM Message Passing**

- All messages have tags
  - an integer to identify the message
  - defined by the user
- Messages are constructed, then sent
  - `pvm_pk(int,char, float) (*var, count, stride)`
  - `pvm_unpk(int,char, float)` to unpack
- All processes are named based on task ids (tids)
  - local/remote processes are the same
- Primary message passing functions
  - `pvm_send(tid, tag)`
  - `pvm_recv(tid, tag)`

**PVM Process Control**

- Creating a process
  - `pvm_spawn(task, argv, flag, where, ntask, tids)`
  - task is name of program to start
  - flag and where provide control of where tasks are started
  - ntask determines how many copies are started
  - program must be installed on each target machine
  - returns number of tasks actually started
- Ending a task
  - `pvm_exit`
  - does not exit the process, just the PVM machine
- Info functions
  - `pvm_mytid()` - get the process task id

**PVM Group Operations**

- Group is the unit of communication
  - a collection of one or more processes
  - processes join group with `pvm_joingroup("<group name>")`
  - each process in the group has a unique id
    - `pvm_gettid("<group name>")`
- Barrier
  - can involve a subset of the processes in the group
  - `pvm_barrier("<group name>", count)`
- Reduction Operations
  - `pvm_reduce( void (*func)(), void *data, int count, int datatype, int msgtag, char *group, int rootinst)`
    - result is returned to rootinst node
    - does not block
  - pre-defined funcs: PvmMin, PvmMax, PvmSum, PvmProduct
PVM Performance Issues

- Messages have to go through PVMD
  - can use direct route option to prevent this problem
- Packing messages
  - semantics imply a copy
  - extra function call to pack messages
- Heterogenous Support
  - information is sent in machine independent format
  - has a short circuit option for known homogenous comm.
    - passes data in native format then

Sample PVM Program

```c
int main(int argc, char **argv) {
  int myGroupNum;
  int friendTid;
  int myid;
  int tids[2];
  int message[MESSAGESIZE];
  int c,i; okSpawn;

  /* Initialize process and spawn if necessary */
  myGroupNum=pvm_jiongroup("ping-pong");
  myid=pvm_mytid();
  if (myGroupNum==0) { /* I am the first process */
    pvm_catchout(stdout);
    okSpawn=pvm_spawn(MYNAME,argv,0,"",1,&friendTid);
    if (okSpawn!=1) {
      printf("Can't spawn a copy of myself!
      pvm_exit();
    exit(1);
  } else { /*I am the second process */
    friendTid=pvm_parent();
    tids[0]=mytid;
    tids[1]=friendTid;
  }

  pvm_barrier("ping-pong",2);

  if (myGroupNum==0) {
    /* Initialize the message */
    for (i=0 ; i<MESSAGESIZE ; ++i) {
      message[i]='1';
    }
    /* Now start passing the message back and forth */
    for (i=0 ; i<ITERATIONS ; ++i) {
      if (myGroupNum==0) {
        pvm_initsend(PvmDataDefault);
        pvm_pkint(message,MESSAGESIZE,1);
        pvm_send(friendTid,msgid);
        pvm_recv(friendTid,msgid);
        pvm_upkint(message,MESSAGESIZE,1);
      } else {
        pvm_recv(friendTid,msgid);
        pvm_upkint(message,MESSAGESIZE,1);
        pvm_spawn(MYNAME,argv,0,"",1,&friendTid);
      }
    }
    pvm_exit();
  exit(0);
```