README

Parallel Computation.

Aim:

- The goal is to implement a mpi system between many beagel bone boards so that the processing of a task can be distributed among them.

Contents:

---

1. UDP socket programming.

Socket: It is an end point in a communication where the data is generated or recieved and is processed.

- UDP socket: We use the function *socket()*. syntax: int socket(int domain, int type, int protocol);
  - `#include <sys/types.h>`
    `#include <sys/socket.h>`
    `int udpsocket = socket(AF_INET,SOCK_DGRAM,0);`
  - the function return the file descriptor on successfully creating a socket, else it return -1 with errno set appropriatley.
- sockaddr:
  - it is evedent that the sockets communicate using the ip addresses. These details are stored in a structure known as SOCKADDR.
  - `struct sockaddr {`
    `unsigned short sa_family;`
    `char sa_data[14];`
    `};`
  - In order to make the contents of this structure more easy to access another structure is used, i.e SOCKADDR_IN.
  - `struct sockaddr_in{`
    `short sin_family;`
    `unsigned short sin_port;`
    `struct in_addr sin_addr;`

```
char sin_zero[8];
};
```

  - Here :
    - sin_family: type of communication (AF_INET - for ipv4 protocol).
    - sin_port: IP port.
    - sin_addr: IP address.
    - sin_zero: Just padding to make it equal to the structure SOCKADDR.
  - This sums up the c data structure that stores the parameters of the socket.
- bind:
  - Once the socket is created and the socaddr has been provided with the suitable values it is then binded together using the *bind()* function.
  - `bind(udpSocket,(struct sockaddr *)&serverAddr, sizeof(serverAddr));`
  - Here the serverAddr is typecast into sockaddr from sockaddr_in.
  - this registers the socket with the specified address so the communication to that socket can be made using the assigned address.
- Communication:

2.Basic layout