Pimpri Chinchwad College of Engineering
Computer Laboratory - IV

Assignment No - B6

1 Aim

Perform concurrent ODD-Even Merge sort using HPC infrastructure (preferably BBB) using Python/Scala/ Java/ C++.

2 Objective

To develop concurrent programming and multi-core programming skills for odd-even merge sort using HPC infrastructure.

3 Software Requirements

- Linux or Windows Operating System
- Gcc Compiler

4 Mathematical Model

S = s, e, x, y, fme, DD, NDD, memshared

S = Initial State
E = End State
X = Input Valueunsorted array

Y = Output sorted array using odd-even merge sort Fme = Main function
DD = Deterministic data
NDD = Non-deterministic data
Memshared = Core that is used for execution i.e. core1/core2 BeagleBone

5 Theory

5.1 Odd-Even Merge Sort

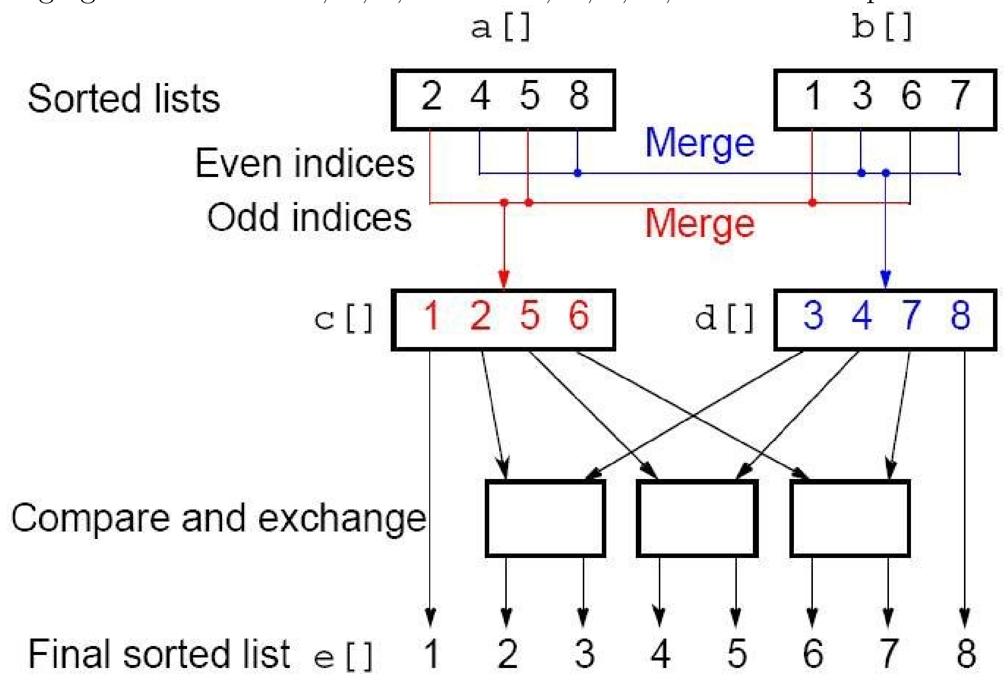
Oddeven sort is a relatively simple sorting algorithm, developed originally for use on parallel processors with local interconnections. It is a comparison sort related to bubble sort, with which it shares many characteristics. It functions by comparing all odd/even indexed pairs of adjacent elements in the list and, if a pair is in the wrong order (the first is larger than the second) the elements are switched. The next step repeats this for even/odd indexed pairs (of adjacent elements). Then it alternates between odd/even and even/odd steps until the list is sorted.

5.2 Merge Sort

Merge sort (also commonly spelled mergesort) is an efficient, general-purpose, comparison -based sorting algorithm. Most implementations produce a stable sort, which means that the implementation preserves the input order of equal elements in the sorted output. Mergesort is a divide and conquer algorithm that was invented by John von Neumann in 1945.

5.3 Example

Merging the two lists a1,a2,...,an and b1,b2,...,bn, where n is a power of 2.



5.4 Beaglebone Black

BeagleBone Black Overview:
The BeagleBone Black is the latest addition to the BeagleBoard.org family and like its predecessors, is designed to address the Open Source Community, early adopters, and anyone interested in a low cost ARM Cortex-A8 based processor. It has been equipped with a minimum set of features to allow the user to experience the power of the processor and is not intended as a full development platform as many of the features and interfaces supplied by the processor are not accessible from the Beagle Bone Black via on board support of some interfaces. It is not a complete product designed to do any particular function. It is a foundation for experimentation and learning how to program the processor and to access the peripherals by the creation of your own software and hardware. It also offers access to many of the interfaces and allows for the use of add-on boards called capes, to add many different combinations of features. A user may also develop their own board or add their own circuitry.

Board Component Locations:
This section describes the key components on the board. It provides infor-

mation on their location and function. Familiarize yourself with the various components on the board.

- Connectors, LEDs, and Switches
- DC Power is the main DC input that accepts 5V power.
- Power Button alerts the processor to initiate the power down sequence.
- 10/100 Ethernet is the connection to the LAN.
- Serial Debug is the serial debug port.
- USB Client is a mini USB connection to a PC that can also power the board.
- BOOT switch can be used to force a boot from the SD card.
- There are four blue LEDS that can be used by the user.
- Reset Button allows the user to reset the processor.
- uSD slot is where a uSD card can be installed.
- microHDMI connector is where the display is connected to.
- USB Host can be connected different USB interfaces such asWi-Fi, BT,Keyboard, etc

Features of Beaglebone Black

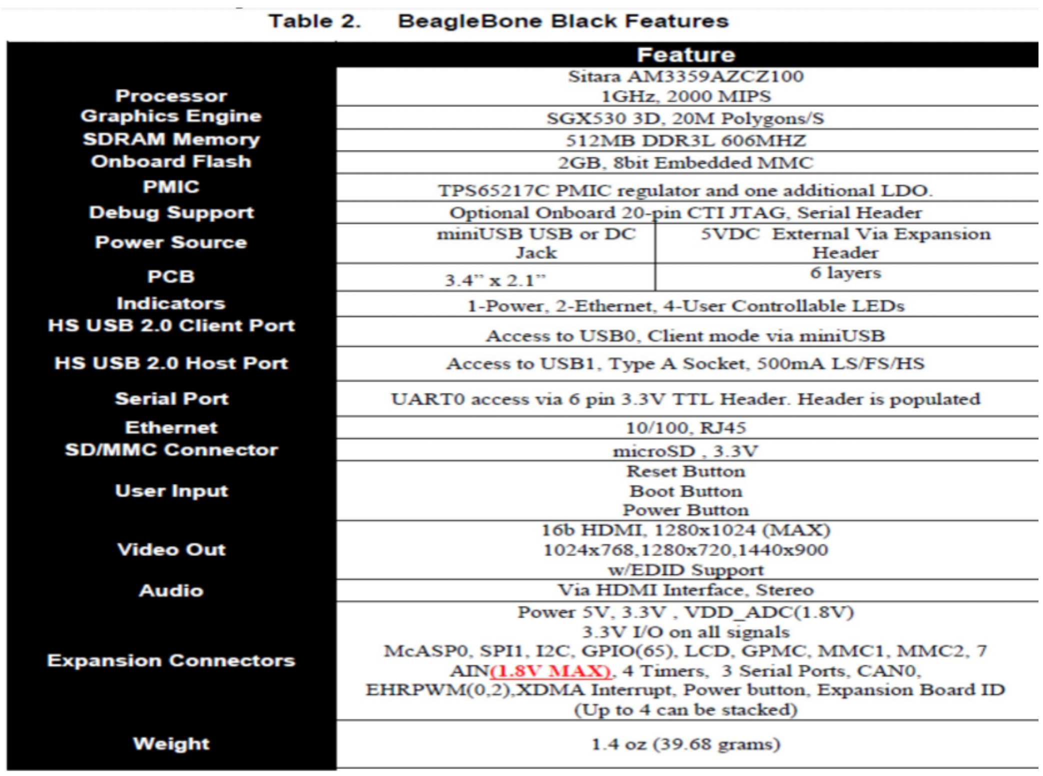


Table 2. BeagleBone Black Features

	Feature	
Processor	Sitara AM3359AZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 606MHZ	
Onboard Flash	2GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(65), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 3 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	

Key Components

- Sitara AM3359AZCZ100 is the processor for the board.
- Micron 512MB DDR3L is the Dual Data Rate RAM memory.
- TPS65217C PMIC provides the power rails to the various components on the board.
- SMSC Ethernet PHY is the physical interface to the network.
- Micron eMMC is an onboard MMC chip that holds up to 2GB of data.
- HDMI Framer provides control for an HDMI or DVI-D display with an adapter.

Connectivity

- Connect the small connector on the USB cable to the board as shown in Figure 1. The connector is on the bottom side of the board.

- Connect the large connector of the USB cable to your PC or laptop USB port.
- The board will power on and the power LED will be on as shown in Figure 2 below.

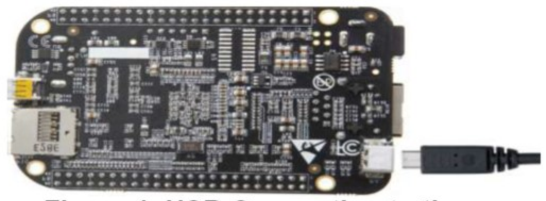


Figure 1. USB Connection to the Board

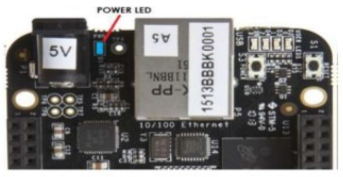


Figure 2. Board Power LED

- Apply Power: The final step is to plug in the DC power supply to the DC power jack as shown in Figure4 below.
- Booting the Board: As soon as the power is applied to the board, it will start the booting up process. When the board starts to boot the LEDs will come on in sequence as shown in Figure 5 below. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it boots the Linux kernel.

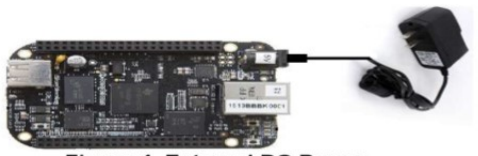


Figure 4. External DC Power

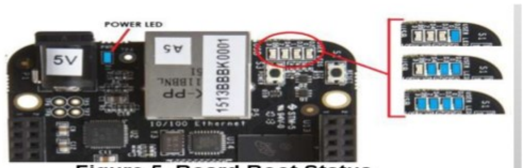


Figure 5. Board Boot Status

5.5 Steps to run program in Beaglebone Black

1. In red hat terminal we need to type following command for accessing the beagle bone terminal. ssh 192.168.7.2
2. Open another redhat terminal for copying the python program from redhat to beagle bone by using command: scp filename.py root@192.168.7.2:
3. Now open beagle bone terminal to check whether program is copied or not by using command ls.
4. To run the program write following command in beagle bone terminal. python filename.cpp

6 Algorithm

6.1 Odd-Even Merge Sort

```
template ¡class T¿
void OddEvenSort (T a[], int n)
for (int i = 0 ; i ¡ n ; i++)
if (i & 1) // 'i' is odd
for (int j = 2 ; j ¡ n ; j += 2)
if (a[j] ¡ a[j-1]) swap (a[j-
1], a[j]) ;

else

for (int j = 1 ; j ¡ n ; j += 2)
if (a[j] ¡ a[j-1])
swap (a[j-1], a[j]) ;
```

6.2 Merge Sort

```
function merge_sort(node head)
if (head == nil) return nil
var node array[32]; initially all nil var
node result
var node next var int i result = head
// merge nodes into array while
(result != nil) next = result.next; result.next = nil
for(i = 0; (i ¡ 32) && (array[i] != nil); i += 1)
result = merge(array[i], result)
array[i] = nil
// do not go past end of array if (i == 32)
i -= 1 array[i] = result
result = next
// merge array into single list
result = nil
```

```
for (i = 0; i ¡ 32; i += 1)
result = merge(array[i], result)
return
result
```

6.3 Parallel Odd Even Sort

```
void ODD-EVEN-PAR(n)

id = process label
for (i= 1; i¡= n; i++)

if (i is odd)
compare-and-exchangemin(id+1);
else
compare-and-exchangemax(id-1);
if (i is even)
compare-andexchange-min(id+1);
else
compare-and-exchange-max(id-1);
```

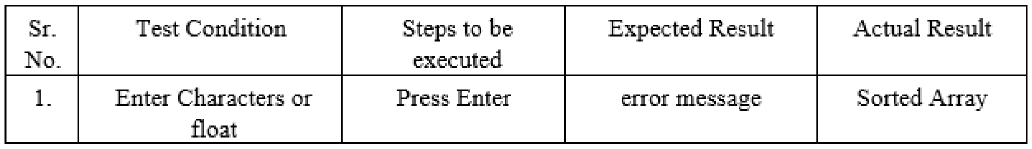
6.4 Complexity

O(log2n)

7 Testing

7.1 Positive Testing

7.2 Negative Testing



Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Enter Characters or float	Press Enter	error message	Sorted Array

8 Conclusion

From this assignment we have studied concept of OBST and developed time and space efficient algorithm using multicore , concurrent environment.

Roll No.	**Name of Student**	**Date of Performance**	**Date of Submission**
302	Abhinav Bakshi	15/3/16	17/3/16