CSE 511 Operating Systems
Project 2 Report

Member 1: Ajay Rahul Pradeep
Email: akp6019@psu.edu
Member 2: Name: Vipul Gupta
Email: vkg5164@psu.edu

Steps to run the program:

- Clone the repository from : https://github.com/PSU-CSE-511/project-2-dsm-and-map-reduce-framework-ajayrahul97
- Distributed Lock - Ricart Agarwala
    - Run 'make' in the dist_lock_ricart folder
    - nodes_list.txt will have the list of IPs and lock numbers
    - Run ./psu_node on each machine
    - This has been tested for 3 machines currently.
- Sequential Consistency
    - Run 'make' in the seq-consistency folder
    - nodes_list.txt will have the list of host addresses.
    - On four different machines run './p1', './p2' and './p3' and './dir' respectively
- Distributed Sorting
    - Run 'make' in the seq-consistency folder
    - nodes_list.txt will have the list of host addresses.
    - Use even number of machines - 2 , 4
    - In each machine run './app1 <process _num> <total_num_processes>'
    - And in the last machine run './dir'
- Word Count (Map-Reduce)
    - Run 'make' in the word_count folder
    - nodes_list.txt will have the list of host addresses.
    - In each machine run './mapreduce <process _num> <total_num_processes>'
    - And in the last machine run './dir'
- K-Means (Map-Reduce)
    - Run 'make' in the seq-consistency folder
    - nodes_list.txt will have the list of host addresses.
    - Use either 2 or 4 machines (since we only have 4 centroids)
    - In each machine run './kmeans <process _num> <total_num_processes>'
    - And in the last machine run './dir'

Distributed Shared Memory Protocol Explanation:

DSM has been implemented using Invalidation based 3-state protocol (Invalid, Read-Only, Read-Write states at each node).

Table maintained at the directory had the rows and columns as mentioned in the below example table.

	Node 1	Node 2	Node 3	RO/RW Status	Mutex Lock	Malloc Region Name
Pg id 1	1	0	0	1	lock_x	"A"
Pg id 2	1	1	1	0	lock_y	"A"
Pg_id 1	1	1	1	0	lock_z	"B"

Every entry is stored as a tuple of pg_id, presence bits vector, ro/rw status bit, mutex lock and malloc name. The table is a vector of such tuples.

Program Flow -

Initialization :

<psu_dsm_system.h> contains the logic for initialization.

- First step is to read the node_list file and store the information of the participating nodes.
- Initialise the Segmentation Fault Handler
- Check if all nodes are ready - Optional - but it was better for coherence.
- Using the start address and the memory size, and the malloc region name, store the page information locally.
- Now for all the pages, change the access to PROT_READ using mprotect.
- Now for all the pages, send a register page request to the directory.
- Directory will create new entries for the page if it doesn't exist, otherwise will update the node's presence bit to 1 if the page is in RO state. Then it will return the acknowledgement.

Write Fault :

- When a node hits a write fault - inside the segment handler, the page id and the malloc name is deduced using the fault address.
- A Write_Page request is sent to the directory.
- If the page is in RW state

        - The directory will first send an RPC call to other nodes to invalidate the page.
        - It will send back the latest page value and an acknowledgment to the node that is requesting the write.
        - The node will first update its page to the latest value.
        - Then the node will change the access to PROT_READ | PROT_WRITE, and exit the fault handler
        - The directory will unlock the page, and change the presence bits accordingly.
- If the page was in RO state
    - The directory will send an invalidation to other nodes.
    - Then directory will simply instruct the node to directly change the rights to PROT_READ | PROT_WRITE without updating the latest value.

Read Fault:

- In case of a read fault, the node will first figure out the page id and the malloc region name. It will send a Read_Page request to the directory.
- If the page is in RW state :
    - Directory will find the node X having the latest value of the page.
    - Request the page value from that node X via RPC.
    - Change the page state to RO and change the presence bits for both X and Y.
    - Return the page to the node Y that had requested the Read.
- If the page is in RO state:
    - Directory will just find the first node having the presence bit as 1.
    - Get the latest value from that node.
    - Change the presence bits
    - Return the page to the node that had requested the Read.
- Once the latest value of the page is obtained, we do a memcpy at the start_address of the page and exit the segmentation fault handler.

Distributed Sorting

A global array of int is shared among nodes, and each node performs partial sorting. Finally the first node merges all the partially sorted numbers. This has been tested with 2 and 4 nodes.

Word-Count

We have developed an extension of the toy example.
An array of key-value struct for all the words in the words in the input file has been shared in the DSM.
It has been assumed that the maximum number of words in a file is 36000.
Function of the mapper : To simply set the value for each word as 1.
Function of reducer: To count the occurrences of 1 for each word and output it in a file. Once the reducer considers the word, it sets the value back to zero, so that the other reducer doesn't pick it up.
The work is divided equally among the nodes.

K-Means

What is shared in the DSM :

- A matrix of x,y coordinates of size 6 X 401
- The first row is used for storing the points. It has been assumed that the maximum no.of points in the input file is 400.
- The second row stores the number of points and the coordinates of the centroids.
- The next four rows store the points belonging to each cluster.

Function of mapper : Mapper calculates the euclidean distance of each point from the four centroids and pushes it to one of the four rows corresponding to the nearest clusters i.e shortest distance from the centroids.

Function of reducer : Calculate the average of all the points in the cluster to calculate the new centroid, and output it in a file.

Distributed Lock - Ricart Agrawala:

- The Ricart Agrawala algorithm has been implemented.
- This has been tested with 3 nodes.
- Each node spends 30 seconds in the critical section and prints a value.
- It was observed that no two nodes were present in the critical section at the same time.

Issues/difficulties Faced:

- Some assumptions have been made for the map-reduce programs with respect to the maximum size of input data
- In all the given programs we have only used psu_register_segment.
- Psu_dsm_malloc tested only for sequential consistency.

References :

[1] GRPC : https://grpc.io/docs/languages/cpp/quickstart/

[2] mProtect : https://man7.org/linux/man-pages/man2/mprotect.2.html

[3] Posix_memalign : https://man7.org/linux/man-pages/man3/posix_memalign.3.html