LOGICAL PARALLELISM IN COMPUTER LANGUAGES

S.MOHAN,.
ASST PROFESSOR IN COMPUTER SCIENCE AND ENGINEERING
V.S.B.COLLEGE OF ENGINEERING TECHNICAL CAMPUS
KINATHUKADAVU-COIMBATORE-642109.
TAMIL NADU. INDIA.

ABSTRACT:

In this paper we will discuss with computer languages used logical parallelism. How to use and how to understand the logical parallelism ? . It means involves structuring code to explore the simultaneously execution , breaking task in to independent sub task for concurrent processing on multi core/ processes. Multi processing enhancing faster performance for the large problems by tasking part of these at once. Contrasting with serial execution . where step depends am previous once .it for an what can run simultaneously task parallelism / data parallelism. Parallelism means actually running the task simultaneously. As different hardware unit concurrently involve the managing multiple task . but they might interleaved an one core through they can also run in the parallel. The instruction level parallelism . the processor execute multiple independent instruction from a single tread at once . the python have the multi processing modules from true parallelism(By passing GIL)- Global Interpreter lock-mutex (multi –executable lock . logical programming declarative model can inherently exposes more parallelism. Example (AND) parallelism (OR) parallelism. The imperative models . When parallelism must be explicitly managed.

Key words: GIL,and, or,mpi.

INTRODUCTION:

Logical parallelism in computer languages have the simultaneously execution of the multiple independent computation improving the performance by breaking down large task in to smaller , concurrent sub task it moves the beyond the serial processing by utilize the multiple processor or core reducing the total execution time for complex algorithms it is crucial for high performance computing can accept and type of logical parallelism .

- Data parallelism
- Task parallelism
- Instruction level parallelism
- It approaching in programming language
- Explicit parallelism

- Implicit parallelism
- extensions

it benefit increase the throughput efficient resources utilize and the ability to solve large and more complex problems .challenges require the careful management of dependency , synchronization and communication between the task. Parallelism is essential for modern computing over coming the limitation of single processor performance by exploiting concurrency .that is (SIMD) single instruction multiple data parallelism shift programming paradigm from strict sequential (linear) logic to concurrent model ,requiring in many case , explicitly management and communication between thread.

C/PYTHON PARALLELISM

To write a program that take advantageous of such easily available parallel computing resources, c and python program have library of their disposal ,such as open MP(C) .and multi processing (python) for shared memory parallel programming and open mpi(c/python for distributed memory parallel processing .

PYTHON:

Concurrency and parallelism related but distinct concept in computer science . concurrency refer to the ability of program handle multiple task at same time .while parallelism refer to the ability if program to execute multiple task simultaneously usually on separate core or processor.

C:

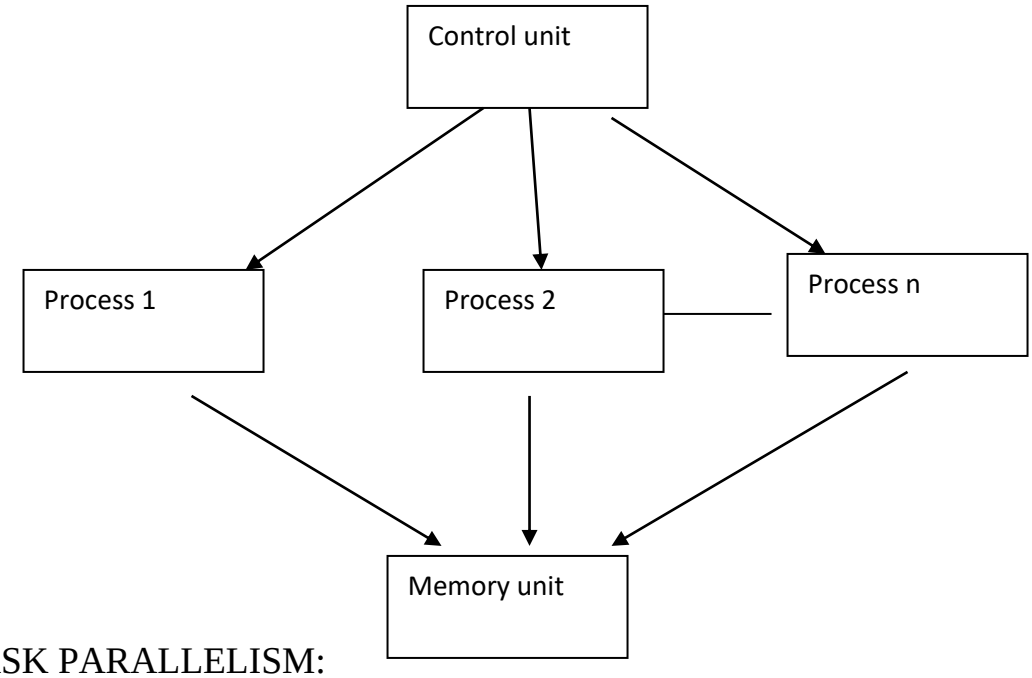
Running the multiple task at the same time with in the system improve the overall performance. The program under execute is known as the process. every program with in the system is independent of each other and execute in isolation with in the own memory space.

Parallel program is a technique that allow multiple computation to be performed simultaneously taking advantageous of multi-core process and distributed computing system. Parallel processing can improve the system performance by dividing the biggest task in to smaller chunk and executing them parallel.

DATA PARALLELISM:

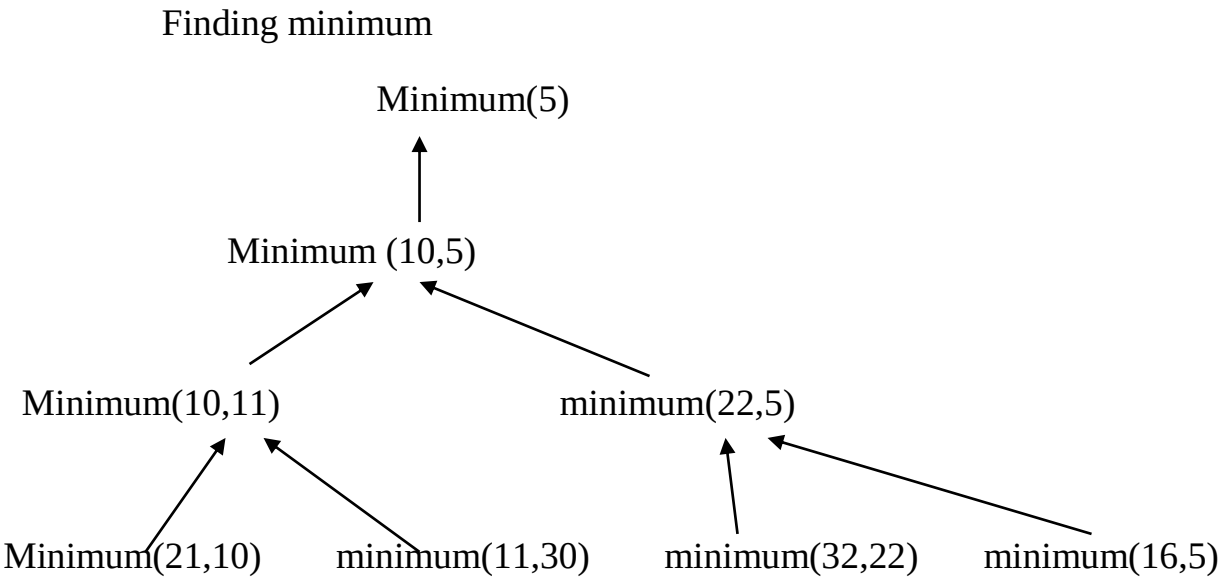
In these parallelism that task need to be carried out are identify fast and mapped to the process. These mapping of the task on to the process is being done statistically or semi-

statistically . the task that is being performed by every process same or identical. But the data on which this operation or task are performed in different.



TASK PARALLELISM:

The inter relation among the task in the task dependency graph can be used for reducing the instruction task solving those problem in which task are associated with the large amount of data as compared to the actual computation . the parallelism that is described with the task dependency graph are each task is know as task parallelism that is describe with task dependency graph where each task is an independent task is known as a task parallelism.



This graph task to fix the minimum of the given stream. Minimum (22,10) and minimum(11,20) compared to and passed an future by one process similarly at same time the minimum(10,5) is calculated and passed an to the further process this approach of computation required less time and less effort.

INSTRUCTION LEVEL PARALLELISM:

Instruction level parallelism is (ILP) a concept and computer architecture that allows multiple instruction to be processed concurrently with in a CPU instruction level parallelism is achieved through technique such as pipe line, super scalar execution and out –of – order execution.

BIT-LEVEL PARALLELISM:

It is form of parallelism computing which based an increasing process . it reduces number of instruction that the system must execute . consider a case where an eight bit processor must add 16 bit integer . the processor from each integer then add to 8 bit lower adder bit from each integer then add to 8 bit higher adder bit, requiring to instruction to compute a single operation a 16 bit processor would be able to compute the operation with single instruction.

Instruction level parallelism must not confused with concurrency ILP a single specific thread of execution of a processor. On the other hand concurrency involves the assignment of multiple thread to CPU core in strict alteration or in tree parallelism if there are enough CPU core ideally one core for each run able thread

Consider the following program

1. s=m+n

2.p=q+r

3.t=s*p

Operation 3 depends on the result of operation 1 and 2 so it can not be calculated unit of both them are completed however operation 1 and 2do depend an any other operation so they can be calculated simultaneously . if we assumed that each operation can be completed in one unit of time then these instruction can be completed in a total of two unit of time giving an ILP of3/2.

LOGICAL PARALLELISM

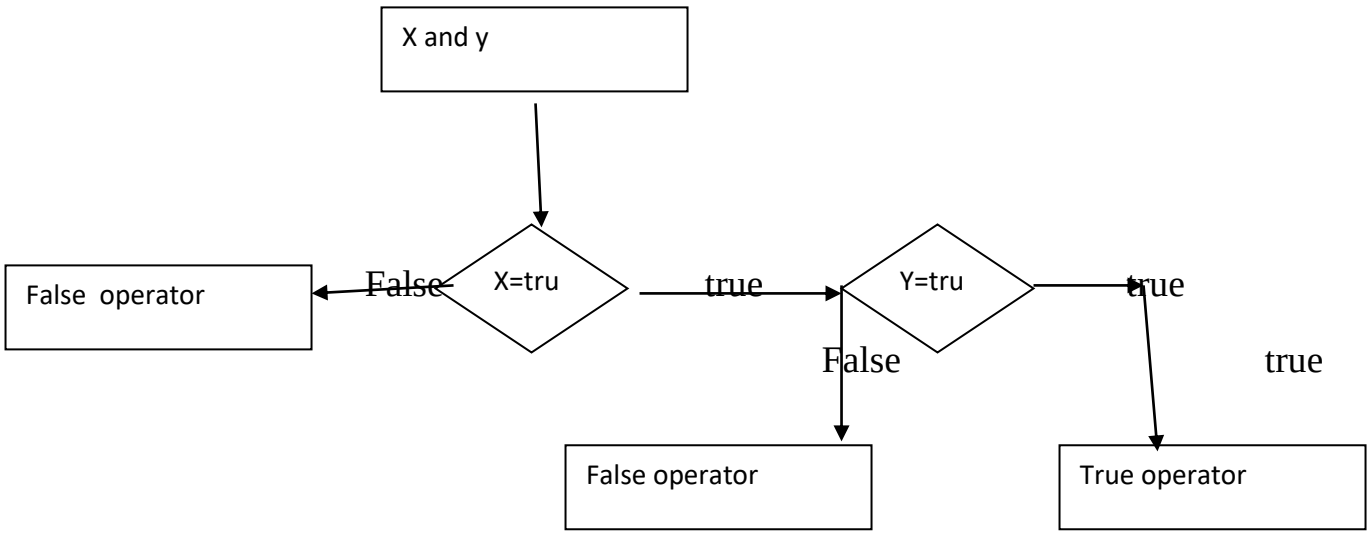
Logical operator are used to combinr or modify condition and return aboolean result (true/false). They are commonly used in conditional structure to control the flow of program based on multiple logical condition.

Truth table for logical operation

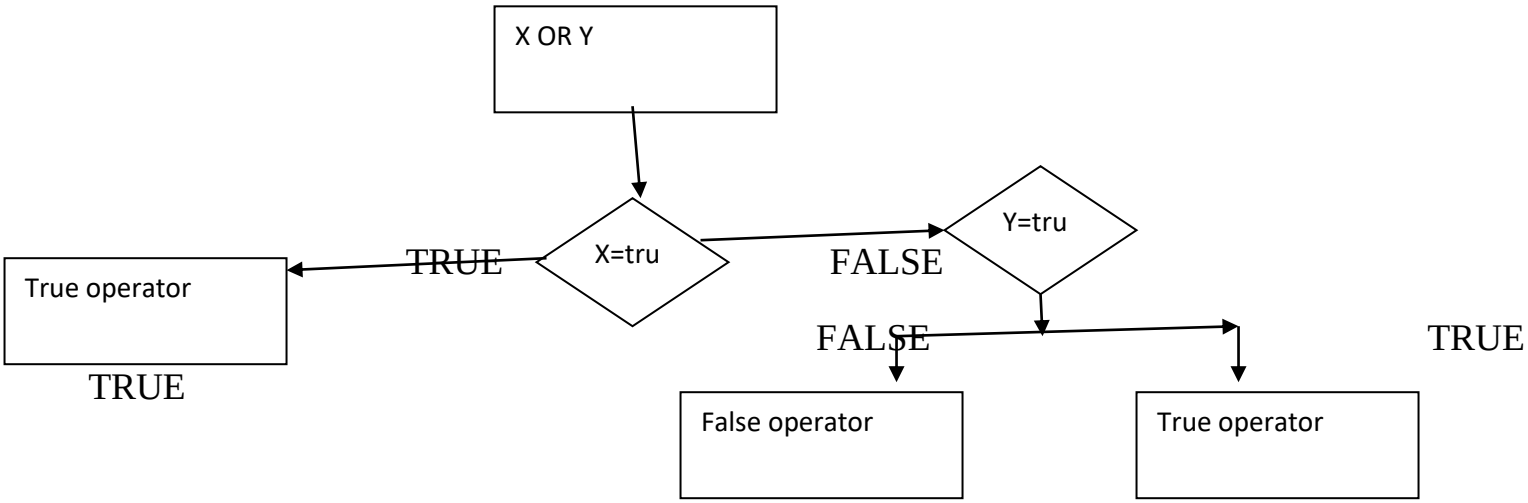
X	Y	X AND y	X or y
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

AND OPERATOR: X * Y (&&) –logical operator

The Boolean AND operator return true if Boolean the operators are true else it is return false.



OR operator: X+Y -(||)- logical operator



CONCLUSION:

Parallel processing and multi-tasking both enhance computing efficiency but differ in execution .multi tasking runs multiple tasks concurrency in fast simultaneously an one or more CPU. While parallel processing execute multiple task simultaneously using multiple processor . a system can be both where it is run multiple application (multi tasking) and uses multiple core to run each application (parallelism). So this is told as logical parallelism in computer languages.

REFERENCES:

1.J. S. Conery, "binding environments for parallel logic programs in non-shared memory multiprocessor", *International Journal Of Parallel Programming*, vol. 17, no. 2, 1988.

2.J. S. Conery and D. F. Kibler, "AND-parallelism in logic programs", *International Conference of Artificial Intelligence*, 1983-Aug.

3.D. De Groot, "Restricted AND-parallelism", *Proceedings of the international conference on Fifth Generation Computer Systems by ICOT*, 1984.

4.F. A. Omara, *Implementation of prolog using massively parallel processors*, 1989.

5.K. A.M. Ali and R. Karlsson, "The MUSE approach to OR-parallel prolog", *International Journal of Parallel Programming*, vol. 19, no. 2, April 1990.

6.K. A.M. Ali and R. Karlsson, Full prolog and scheduling OR-parallelism in MUSE, Sweden, Kista:Swedish Institute of Computer Science, May 1991.

7.Z. Somogyi, K. R. Ohanarao and J. Vaghani, "A backtracking algorithm for the stream AND-parallel execution of logic programs", *International Journal of Parallel Programming*, vol. 17, no. 3, 1988.

8.D. H.D. Warren, *OR-parallel execution models of prolog*, 1987.

9.T. P. Dobry, A high performance architecture for prolog, Kluwer Academic Publishers, 1990.

10.A. Ciepielewski, B. Hausman and S. Haridi, Initial evaluation of a virtual machine for OR-parallel execution of logic programs, Sweeden, Stockholm:The Royal Institute of Technology, department of Computer Systems, 1988.

11.B. Ramkumar, *Machine independent AND and OR parallel execution of logic programs*, 1991.