PAPER SUMMARY

Branch-and-bound is a technique to find the optimal solution to a combinatorial optimization problem by reducing the set of feasible solutions. In travelling salesman problem, the space of feasible solutions can be progressively partitioned, forming a search tree. The number of feasible solutions increases exponentially as a function of the size of the problem input. It can be seen that as the size of the partial tour increases, the number of full tours containing the partial tour decreases.

Distributed branch-and-bound has been widely studied. The first distributed fault-tolerant branch and bound was presented by *Yahofufi* and *Dowaji*. *Parallel Utility for Branch-and-Bound (PUBB)* , written in C introduced the notion of *Logical Computing Unit*. This paper leverages *JICOS* - a Java-centric network computing service that supports high-performance parallel computing to solve the computationally intractable travelling salesman problem using branch-and-bound.

JICOS supports scalable and adaptively parallel computation, tolerates basic faults and hides communication latencies. It consists of three service component classes viz.

- *Hosting Service Provider (HSP)* : This is the one-stop interface for clients where the client logs in, submits tasks, requests results and logs out. The *HSP* also manages task servers explained below.
- *Task Server* : This is a store of Task objects along with a balancing mechanism for ready tasks. It also includes a fault-tolerant mechanism to reassign tasks to different hosts during failures.
- *Host* : This is a compute node that may dynamically join a task server and request execution of tasks.

JICOS also includes a simple set of application-directives for improving performance viz. task caching, task pre-fetching, task server computation. The computations are modeled using a Directed Acyclic Graph (DAG) whose nodes represent tasks. Each task has handles to an immutable input object and a mutable asynchronously shared object.

Classes that comprise the *JICOS* framework assume that the branch and bound problem is formulated as a minimization problem and the cost can be represented as an integer.

Each task corresponding to nodes in a search tree decomposes itself into subtasks until a permissible level of recursion has been reached, or if the subtask becomes *atomic* i.e. small enough to be explored on a single compute server. The cost of the best-known solution at any point of time is shared among tasks by encapsulation in the branch-and-bound computation's shared object. When a branch-and-bound tasks finds a complete solution whose cost is less than the current least cost solution, it sets the shared object to this new value. *JICOS* propogates the new least cost throughout the system.

The above solution for TSP using JICOS was tested on a linux cluster consisting of 1 head machine and 64 compute machines. The problem instance was a 200 city euclidean problem where the initial upper bound for the minimal-length tour was set to the optimum tour length to ensure that the speedup cannot be super-linear. The problem instance decomposes into exactly 61,295 *BranchAndBound* task whose average execution time was 2.05 seconds, and exactly 30,647 *MinSolution* tasks whose average execution time was less than 1 millisecond. The critical path for the problem instance was 37 seconds. The actual fraction of perfect speedup exceeds 0.94 in a normal scenario and it exceeds 0.96, when using an appropriate

number of task servers. The overhead of running a task server on a machine shared with a compute server, was found to be 3115.1 seconds on an average for a dedicated task server and 3114.8 seconds for the shared case. Both of these represent 99.7% ideal speedup.