Solving Busy Beaver Problem using Farmer/Worker Scheme in the context of Worldwide Computer

Boleslaw Szymanski
Bram van Heuveln
Carlos Varela
Kyle Ross
Owen Kellett
Selmer Bringsjord
Shailesh Kelkar

Abstract

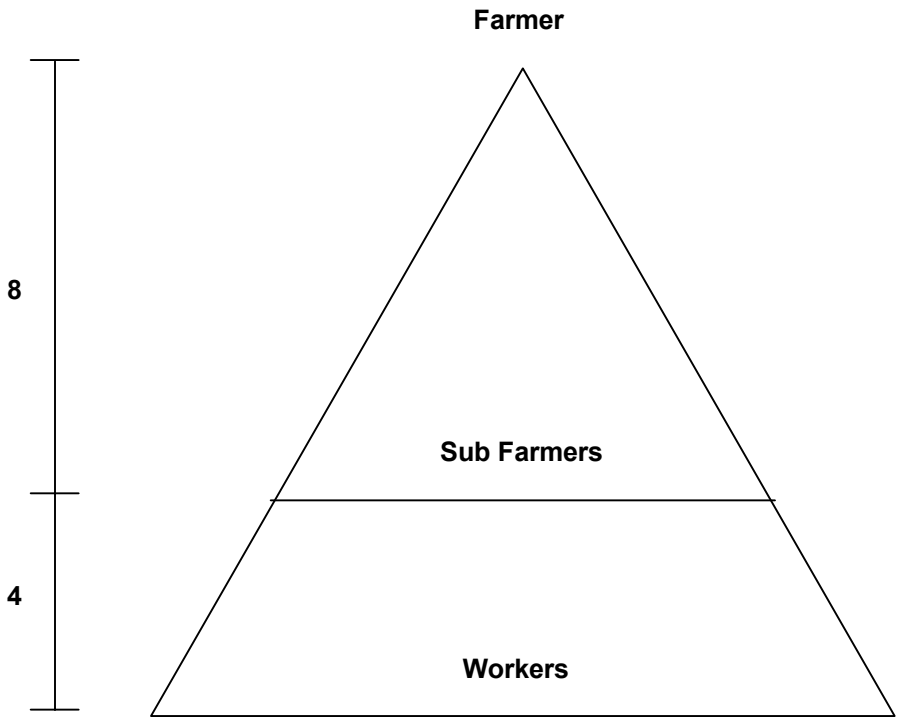
In this paper we present the aspects of solving complex problems using Worldwide Computing based distributed approach. The Busy Beaver Problem of distributed Turing Machine emulation is considered as a running example.

Keywords

Worldwide Computer Distributed Computing, SALSA, Actors, Busy Beaver Problem, Divide-and-Conquer, Load Balancing.

1. **Introduction:** Worldwide Computer enables us to solve complex, computing intensive problems using distributed approach by providing an Internet based infrastructure and an actor oriented programming language, SALSA[1]. The Busy Beaver Problem[2] is to find out the maximum number 1's that can be printed out by an n-state Turing Machine. The enormity of the search space that is involved can be tackled by a distributed approach, using Tree Normalization[3] in the Farmer/Worker Scheme[4]. This paper discusses the various aspects in solving the Busy Beaver Problem by Farmer/Worker Scheme using Worldwide Computer.

2. **Distributed Approach:** The distributed approach to the Busy Beaver Problem involves distributing the problem set over multiple computers using the Farmer/Worker Model in a hierarchical manner. The problem involves generating the Turing Machines from the original Turing Machine at level 0 and processing in a Breadth First manner down the hierarchy till level 8, at which it will use the Farmer/Worker model to distribute the generated tree to different workers. Further refinement of the scheme will have nodes that are workers for the original farmer but also farmers in their own right for the workers at lower level in the hierarchy. Thus they will act as proxy farmer for the workers down the hierarchy. The farmer will be entrusted with the work on generation and normalization of the part of the tree assigned to it, using its computation power and that of the workers at its disposal. We will discuss some of the relevant aspects of this approach in detail.



2.1 Machines Types: The approach should allow for the acceptance of both fully and partially defined machines.

2.2 Generation and Normalization: The generation and normalization of Turing Machines in the tree are highly interrelated and their separation needs to be cleanly defined in the Farmer/Worker scheme.

2.3 Storage: The Turing Machines that need to be considered will be stored at different levels of farmers in the tree. The workers will be concerned with only the queue at the level of its farmer; with the farmer itself can be a worker for another farmer up the hierarchy. This will be useful in the following contexts:

2.3.1: Network Optimizations: At the physical or machine level we will consider a local network with multiple workers reporting to a Farmer in that network. This Farmer will be responsible for communicating with the Farmer up the hierarchy in logical level, which can be a machine in a different network in the physical level. If the network is big we can have multiple farmers in the network reporting to a Farmer up the hierarchy in the same network. This will help in enhancing the communication and better performance.

2.3.2: Firewall Issues: Most of the modern networks have hardware and or software firewalls, which will create problems. The Network Optimizations will enable us to tackle this issue.

2.4: Reconfiguration: The major advantages provided by the Worldwide Computing are the support for distribution and aspects of system reconfiguration. In the Worldwide Computing various machines will join the system randomly and leave randomly except for some dedicated machines which will be used as farmer/s at the upper level/s in the hierarchy. This presents interesting possibilities and challenges:

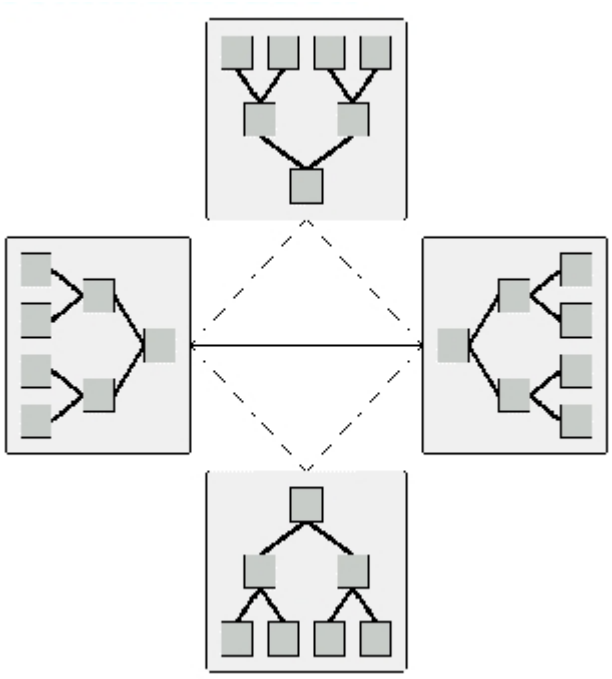
2.4.1: Work Estimation: If we consider the queue at each farmer as the work to be done at the farmer level, then there has to be a generic way of quantifying the work other than the obvious size of the queue. This is important as we are concerned with the different amount of further processing required on each member of the queue. Also it has to generic enough to be applicable while solving problems other than Busy Beaver using this system. For Busy Beaver Problem, the hierarchical refinement of nodes by generating and pruning of Turing machines allows us to refine the estimate of work according to the number of machines pruned at each step. This estimate can be used to do "smart work stealing" using priority queues. Thus the work needs to be quantified by assigning some weight values to the queue members and considering the total of the queue size and the member weights. This will also help in assigning the newly available member to an existing or new Farmer.

2.4.2: Physical Locations: The reconfiguration will also have to consider the physical location of the newly available member as communication times between the worker and its farmer is of vital importance in the context of the system efficiency. The Worldwide Computing uses Universal Actor Names[5] (UAN). UAN are globally unique identifiers, which persist over the life-time of an actor and provide an authoritative answer regarding the actor's current location. The Uniform Actor Locators (UAL) uniformly represents the current location of an actor – called the Theater. As this gives the physical location of the actor we can use this information to assign a farmer that is physically close to the newly available theater. It should be noted that along with the physical location the actual communication times are equally important. This also potentially handles the Firewall and network optimizations issues.

2.4.3: Member Performance Quantifications: In a distributed environment, members will perform at various efficiencies depending on the hardware and communication setup. For an efficient distribution and processing of the work, we need to quantify these performances so as to allow the system to identify good performers and dynamically allocate more work to them.

2.4.4: Actor Migration: The quantification of the member performance discussed above is important in the context of Worldwide Computing, in order to decide on the migration policies of the Actors. The Worldwide Computing also gives the facility of group migration of related actors to improve performance. This is also important in the context of the work stealing approach discussed ahead.

2.5: Work Stealing: In the context of the discussion above regarding quantifying work, member performance and migration as part of reconfiguration, work stealing is a significant in terms of performance and load balancing. Due to dynamic and reconfigurable nature of the system, it is difficult for the programmer to manually optimize applications for hierarchical systems. Cluster aware Hierarchical Stealing (CHS) [6] based on Satin system approach can be used to address these issues. The divide and conquer approach to solve Busy Beaver problem using Farmer/Worker scheme, lends itself well for a hierarchically structured systems, as this leads to a hierarchically structured task graph which can be executed with excellent communication locality. CRS adapts itself to network conditions and job granularities, and does not require manually-tuned parameters.



The CHS in context of our system can be utilized in two ways:

2.5.1: Existing Members: In CHS, when a node is idle, it first asks its child node for work. If the children are also idle, the steal messages will recursively descend the tree. Only when the entire subtree is idle, messages will be sent upwards in the tree, asking parent nodes for work. Thus in the system illustrated in the above figure, only when the cluster root node finds its own cluster node to be idle, it sends a steal message to the root of another, randomly selected cluster. This ensures much locality, as the work inside a subtree will completed before the load-balancing messages are sent to the parent of the subtree.

2.5.2: New Members: When a new member shows up and request work, based on the discussions in the network optimizations and physical location discussed, above the CHS becomes applicable. The further refinement that we can do is to utilize the information regarding the member performance as discussed above in member performance quantification to assign work from a farmer, sub farmer or worker depending on its performance.

2.5.3: Starvation: In some scenarios, the actors may not migrate from one theater to another as the current theater may be performing well. The newly created theaters or existing theaters which have completed work might be starved of work. These cases need to be handled by avoiding starvation, by forcing at least one or more actors to migrate to newly created theater or existing theater with no actors in order to get better load balancing.

3: Issues: The system relies on the top level farmer for getting and processing final results. So the issue of failure at top level needs to be considered. Replication should be used for recovery. The moving of more work down the tree also introduces the possibilities of failures of sub farmers which may not be under our direct control unlike the root farmer.

4: References:

[1] Carlos Varela and Gul Agha: “Programming Dynamically Reconfigurable Open Systems with SALSA”.

[2] T. Rado: “On non-computable functions”. The Bell System Technical Journal, 41(3): 877-884, 1962

[3] B.van Heuveln, et al. “Attacking the Busy Beaver Problem by incorporating the Tree Normalization Method into a Farmer/Worker Scheme”.

[4] Boleslaw K. Szymanski et al. “Twin Primes Enumeration”, http://www.cs.rpi.edu/research/twinp/

[5] Carlos Varela and Gul Agha: “Naming and Mobility in Worldwide Computing”.

[6] Rob V. van Nieuwpoort et al. “Efficient Load Balancing for Wide-Area Divide-and-Conquer Applications.