A Distributed Parallel Processing Environment Based upon the Linda Paradigm: A Research Prospectus

Kenneth D. Landry, George Cline, and James D. Arthur

TR 92-18

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

April 15, 1992

A Distributed Parallel Processing Environment Based Upon The Linda Paradigm: A Research Prospectus

Kenneth D. Landry
George Cline
James D. Arthur

Abstract

As we enter the computing era of the nineties, problems are becoming increasingly more complex and time consuming. As the computing capacity of the uniprocessor is being taxed, and the high cost of parallel and super-computers is still prevalent, alternative methods of achieving parallel performance at an economical price is desired. This proposed research effort offers one such alternative, focusing on the idle CPU cycles existing on local area networks. With the increase in the computing power of workstations and their declining costs, one can effectively transform the unused computing power attached to a local area network into a parallel processing environment. Effectively exploiting such an environment, however, requires a specification and operational framework that is portable, easy to use, and efficient. The environment is constructed around the Linda parallel programming paradigm which provides an effective parallel computational framework.

I. Introduction

As we enter the computing era of the nineties, problems are becoming increasingly more complex and time consuming. For example, mapping out the human genome requires a significant amount of computing power and time to analyze the approximately 100,000 genes in the human cell, as well as developing physical maps of each chromosome and determining the sequences of various DNA chains. Approximately 15 years and 3 billion dollars is required to determine the make up of the human genome. Powerful parallel computers, vector processors and special-purpose computers are a necessity. However, the expense of such machines far exceeds the limits of many users.

As the computing capacity of the uniprocessor is being taxed, and the high cost of parallel and super-computers is still prevalent, alternative methods of achieving parallel performance at an economical price is desired. This proposed research effort offers one such alternative, focusing on the idle CPU cycles existing on local area networks. With the increase in the computing power of workstations and their

declining costs, one can effectively transform the unused computing power attached to a local area network into a parallel processing environment. Effectively exploiting such an environment, however, requires a specification and operational framework that is portable, easy to use, and efficient.

The research direction being followed by the Linda group at Virginia Polytechnic Institute focuses on the development of a parallel processing environment utilizing a local area network with low cost personal computers. The environment is constructed around the Linda parallel programming paradigm [CARRI89a, GELER85a and GELER85b] which provides an effective parallel computational framework. In particular, Linda

- establishes a coordination language which allows a parallel program to perform process creation, synchronization, inter-communication and the sharing of distributed data structures through a logically shared object memory, called tuple space[BERND89],
- reduces the programmer's onerous task of dealing with the simultaneities of multiple executing processes within a parallel program, by allowing the programmer to develop each executing process independently from the rest,
- supports parallelism through replication as well as partitioning[1], and
- is portable; implementations of Linda systems exist on several different machine architectures and network configurations[2].

Building on the basic Linda paradigm, additional advantages are offered by the proposed implementation of Linda-LAN[3]:

- A *true* parallel computational environment is established.
- *Idle CPU cycles* existing on local area networks are exploited.
- An architecturally *low cost* parallel environment is made accessible to a wide range of users.
- An internal *visualization* of executing parallel computations are provided to the user via a set of constructed tools.

---

[1] While many parallel methodologies require a task to be partitioned into distinct and disjoint sub-tasks which execute in parallel, Linda provides the added capability of replication. That is, a specific task can be duplicated many times, and each instance simultaneously operates on a different set of data.

[2] Implementations include the Sequent and Encore multi-processor shared memory machines as well as VAX/VMS Ethernet networks.

[3] Linda-LAN is a software-based framework. The LAN is the only "specialized" hardware needed to support the proposed parallel environment.

Although the proposed Linda-LAN touts a desirable parallel specification framework at an economical price, computational power is also a requirement. Without achieving high levels of performance, problems cannot be solved in a timely manner. Linda networks at Sandia National Laboratories have out-performed the Cray 1S and a single processor of the Cray XMP on production code for parameter-sensitivity analysis; the implementations, however, were not full-scale Linda systems[WHITE88]. The proposed research centers on developing a complete *distributed Linda parallel processing environment on a common local area network* that attains significant performance. As stated below, the objectives of this research are divided into the short and long term goals.

Short Term Goals:

(1) Develop a distributed version of Linda-LAN with access to a central data repository, or tuple space, through a dedicated workstation, e.g. a tuple server.

(2) Create tools to analyze performance and to provide internal views of Linda-LAN.

(3) Identify performance and language deficiencies, and correct them.

Long Term Goals:

(1) Distribute tuple space throughout the Linda-LAN.

(2) Upgrade the performance and reliability of the Linda-LAN to an acceptable level.

(3) Supply improvements to the Linda language, such as (a) enabling separately executing parallel programs to securely share information through a multiple-level tuple space and (b) handling deadlock within a Linda program.

(4) Build a powerful tool which dynamically monitors the performance of the Linda-LAN and intelligently load-balances the Linda system.

Once the short term objectives have been reached, major benefits of the proposed research should be realized. In particular, the improved computing power and increased problem base of the Linda-LAN over a single processor of the network should be evident. Additionally, the cost savings of the Linda-LAN should be recognized. The long term objectives should cultivate parallel programming and establish the Linda-LAN as an effective parallel programming environment at an economical price.

This research proposal begins by giving a brief background of the Linda paradigm and discussing the current research efforts within the VPI Linda group. A presentation of the proposed work in building a Linda parallel environment on a local area network within the short and long term framework is described, as well as the inherent problems of distributing Linda to a network.

II. Background

Linda is a coordination language rather than a complete parallel programming language. A coordination language [CARRI90 and ZENIT90] provides the primitives to create processes as well as coordinate communication among processes. By virtue of being a coordination language, Linda primitives can be introduced into many base programming languages. The original implementation, using C as its base language, exploits a preprocessor approach which transforms Linda operations into C source code. Implementing the Linda primitives in this way is especially useful when parallel systems need to be developed in multiple languages. Linda has been embedded in a wide variety of other languages - C++, FORTRAN, various Lisps, PostScript, Joyce, Modula-2 and soon Ada [GELER90].

The Linda approach supports process creation and inter-communication through a shared data/process repository called Tuple Space (TS). Linda provides operations to generate data tuples (`out`), to read data tuples (`rd`), and to remove them from TS (`in`). Tuple Space not only contains data tuples but also process tuples (created with the `eval` operation) which are often called "live tuples." These process tuples are instantiated and are eventually replaced by a data tuple when the instantiated process finishes executing. TS can also be used to share data structures among processes and synchronize the order of actions that processes perform.

A number of Linda applications have been written either as examples of Linda's expressivity or as "real" programs used in industry. These applications include DNA sequencing, finding primes and process lattices [CARRI88]. Moreover, Cogent, for its XTM machine, has designed a Unix operating system that reflects the Linda paradigm [LELER90] and has tested the system by developing a Linda-based parallel window server called PIX [LELER88].

When one discusses the Linda paradigm, two characteristics are often touted: its ease of use and its portability. From a conceptual standpoint, parallel programming within the Linda framework is intentionally high level, which is exactly why it is so flexible and powerful. Not so surprising, however, this high-level approach is at the root of Linda's greatest criticism - its questionable performance [DAVID89]. Linda does exhibit acceptable performance on both shared and distributed memory parallel (MIMD) machines [BJORN88, BJORN89 and CARRI87], performance suffers for full-scale local area network platforms.

Because Linda does not rely on any specific type of architecture, Linda programs are easily ported to a wide variety of machines with little or no modification. Machines currently hosting Linda include

workstations such as Sun, DEC, Apple Mac II and Commodore AMIGA 3000UX. Linda has also been ported to parallel machines such as the Sequent, S/Net and the Hypercube.

Many comparisons have been made between Linda and other parallel models of computation. In particular, Gelernter compares Linda to a number of different paradigms like concurrent logic programming (CLP), concurrent object oriented programming and functional programming [CARRI89a]. Subsequent discussions [DAVID89, KAHN89 and SHAPI89] by defenders of each paradigm provided laudatory remarks as well as criticism of Linda's approach.

Regardless of the excitement, however, Linda's main criticism is that of its performance. In defense of Linda, no system is ever "fast enough," especially when one considers the ever increasing computational demands made by programs and programmers today. Nonetheless, speed is a concern, and even more so when one considers placing Linda on a local area network platform. Performance will suffer if Linda is blindly moved to a LAN platform without any consideration for the properties affecting performance inherent in that platform. Attempting to identify and solve such performance issues is one of the focuses of the Linda group at VPI.

To date we have ported Linda to Apple's Mac II and to Commodore's Amiga 3000 UNIX workstations. Our current research objectives are two fold:

- Build an initial version of Linda-LAN.
- Identify performance and/or language deficiencies and take steps toward correcting them.

The remainder of this section elaborates on these two objectives.

In the original version of Linda, the kernel (the set of routines supporting the Linda operations) is attached to each Linda program when started. By having the kernel attached to the Linda program (see Figure 1), the kernel becomes a part of each process that gets spawned off by an `eval` operation. Tuple Space resides in shared memory and is accessed by all processes through their own copy of the Linda kernel routines. The synchronization of kernel access to Tuple Space is achieved through the use of semaphores. The approach of using multiple kernels (attached to Linda processes) accessing TS in shared memory works well for a single machine, but is inappropriate for a LAN platform.

The first step toward a distributed version of Linda has been to isolate the kernel routines and make it a separate process through which all processes access TS. This independent kernel handles all communication among Linda processes through the use of sockets and is also capable of concurrently

managing Tuple Spaces from multiple Linda programs. Our eventual goal is to distribute Linda processes to available workstations on a LAN. Each of these workstations will have a dedicated Linda kernel to field TS requests from the distributed processes. Effectively, a two layer system is desired where the first layer of communication is between Linda processes and the Linda kernel on the same machine, and the second layer is between machine kernels across the LAN (see Figure 2).

Because Linda is being placed on a network and performance is our primary concern, the Linda group is also working on improving the performance of Linda programs. Many of the identified optimizations and changes can be applied to non-network versions of Linda as well. In particular, TS access is a principle bottleneck in the Linda approach; current research efforts focus on alternate storage organizations and access mechanisms to improve overall performance. Moreover, improved compile time and run time analysis is expected to further expedite process communication. One strategy to increase the performance of TS access is to reduce the contents of TS. More specifically, performance can be significantly increased by either physically reducing what is placed into TS or by giving the "illusion" that there are fewer tuples in TS through the use of code motion. In addition, improvements to the language such as providing for multiple tuples spaces are being considered as well as appropriate ways of handling deadlock in Linda.

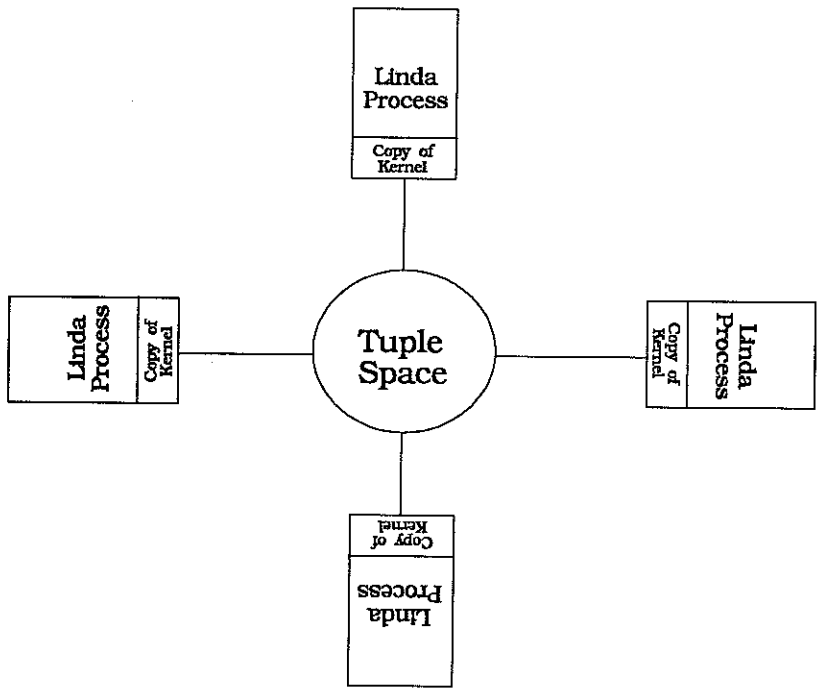

Figure 1. Tuple space access in the original version of Linda.

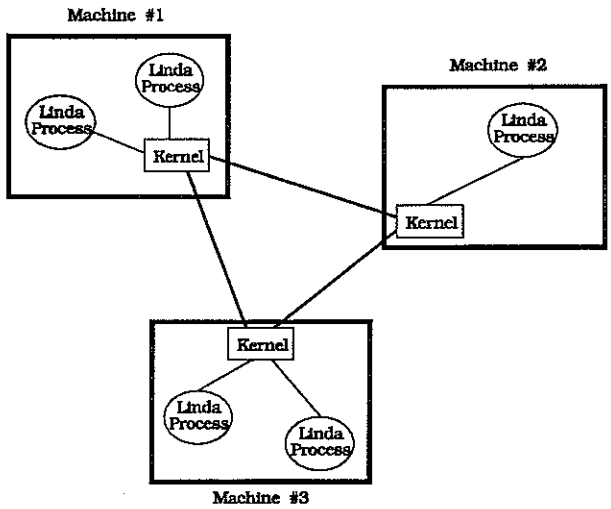

Figure 2. Communication patterns for a typical Linda program running on distributed machines.

III. Proposed Research

The research proposed by the Linda group at Virginia Tech builds a parallel programming environment using a local area network. The environment is based upon the Linda parallel programming paradigm which has a number of advantages; primarily it is a software system requiring no specialized hardware. Moreover, existing machines on a LAN can be used for the environment platform making this proposed parallel processing system a very cost effective solution. This system, however, is not without its problems - namely its inadequate performance primarily due to slow tuple space access. This poor performance may be accentuated when tuples are required to travel over a network. The proposed research can be categorized into 3 areas:

- Distribution of the Linda system on a LAN platform,
- Improvements on the language as well as the performance, and
- Provision of tools for the parallel environment.

The research effort is divided into two phases, both of which involve goals from each category; the phases are the short (1-2 years) and the long (3-5 years) term plans. In the short term plan, the goals include a distributed Linda system with a dedicated tuple space machine (TS server), improved TS access times, and the creation of a monitoring tool to capture system performance statistics and provide an internal visualization of executing processes. The long term plan includes such goals as distributing tuple

space across the network to improve performance, improved tools, adding the capability of multiple tuple space to the Linda language and providing for the detection of system deadlock.

Short Term Plan

The proposed short term plan concentrates on the distribution of the Linda system onto a LAN platform and on the improvement of its performance. However, the construction of tools is also an important part of the short term plan; a monitor tool to analyze the system now and in the future is to be built. In the first phase, the research effort is to achieve the following goals:

1) Distribute Linda on a LAN with TS dedicated on one machine (TS server).
2) Increase system performance by maximizing tuple throughput in TS.
3) Develop a Linda monitor to collect performance statistics.
4) Enhance the Linda monitor to give a visualization of tuple/process activity.
5) Develop Linda applications to show the power of the parallel processing environment.

The first item is the most important goal of the short term plan. The construction of the system is based upon a dedicated tuple space server. This server resides on a single machine and is responsible for fielding all tuple space requests. This system also has a communications server called the Linda manager. Its responsibilities include the determination of which machines are available for participation in computation. This means that machines can dynamically excuse themselves from involvement if the local workload increases beyond a certain threshold. In addition, the manager controls of the amount of network communication the system should utilize.

One of the problems with this distributed design is the bottleneck of tuple space. To alleviate this problem, performance improvements are needed with respect to TS access. These improvements take a number of different forms:

- Improved TS data structures,
- Improved TS access methods based upon new data structures,
- Early initiation of TS requests,
- Implementation of Tuple Futures, and
- Improved process/kernel communication.

Current TS data structures include hash tables in its basic form; one optimization is to enhance the hashing organization to take advantage of "locality of keys." This new hashing scheme, called *segmented hashing*, provides quick access times as well as low space overhead. Improvements in the area of access

methods are possible with *segmented hashing* since the need for linear searching is eliminated (unlike in the current implementation).

Another improvement is in recognizing that speedup can be achieved if a request for a tuple is initiated in advance of when it is actually needed. A similar optimization is not to block when requesting a tuple (on an `in`) but to simply continue on until data from the tuple is actually needed - "tuple futures." By the time the data is needed, the request for the tuple could already be filled.

The last improvement increases performance by adding specialized communication primitives between the process and the kernel. In an attempt to keep the language small, these primitives are not available to the user but are invoked by the compiler when opportunities to use them are recognized within the code. For example, it is common in Linda programs to have a tuple similar to ("Counter", 5). This counter tuple is often incremented with an `in`("Counter", ?i) followed by an `out`("Counter, ++i). By having the Linda compiler recognize this occurrence, a special primitive could be used to tell the kernel to increment the counter "in place" and return its value. These types of optimizations can significantly reduce the amount of communication overhead needed between the process and the kernel.

In order to determine the significance of the performance improvements, special tools are required to collect statistical data on each new enhancement to the system. As part of the short term objectives, a Linda-LAN performance monitor is to be constructed. Each Linda parallel program is traced while executing on the system. Data on tuple space access, kernel routines, and Linda processes is gathered and reported at program termination. Comparisons can then be made concerning the enhancements. In addition, detailed information regarding tuple working sets, workstation performance, communication delays, tuple space blockages, and process blockages is offered. This information is essential in providing more insights into improving system performance.

The unique feature of the Linda-LAN monitor is its ability to track a program across the entire network. Individual workstations are interrogated as well as the program execution as a whole. The monitor also provides a dynamic view of the program as it is executing. Process instantiations, terminations, synchronizations, and data dependencies are displayed through a graphical user interface. The internal visualization produced by the monitor gives the programmer and system developers a better understanding of the behavior of the program and the system. Equipped with the knowledge of how the program operates, the programmer is able to tune the program to the current environment. System developers gain important information regarding system bottlenecks.

The final objective of the short term plan is to develop a test-bed of applications which can effectively take advantage of the computing resources offered by Linda-LAN. For instance, parallel database queries can be made via multiple Linda processes executing on various workstations of Linda-LAN. These processes may perform pre-processing and post-processing of requests to a database. Once the data has been processed, the information can be relayed via tuple space to another Linda process for display to the user. Other areas of interest include simulation studies, graphics, and multimedia.

Long Term Plan

The proposed long term plan is intended to concentrate on improvements to the system. These improvements range from changes to the implementation to changes to the language. The following is a list of the goals for this phase of the research.

1) Distribute Tuple Space across the network.
2) Build in better reliability and fault tolerance in the system.
3) Add the language capability to declare multiple tuple spaces.
4) Provide for the ability to detect system deadlock.
5) Continue to provide system tools (e.g. a monitor) and improve system performance.

In order to continue to improve system performance it would be desirable to have TS distributed across the network instead of on a single machine. One approach to distributing tuple space is to partition it into disjoint pieces and place each piece on a separate machine. This parallelizes the work that the distributed TS manager has to do. Another approach is to duplicate tuple space on several machines. By doing this, a process sending or requesting a tuple can access the closest TS machine. All copies of tuple space have to be kept in sync (which could possibly be expensive) but this offers reliability and fault tolerance to the system if a TS machine should go down. It has been discussed in the literature that one of the deficiencies of the Linda language is its inability to declare multiple tuple spaces [KAHN89]. This capability is one of the goals of the long term plan. In terms of robustness, the ability to detect when a Linda program (or part of the program) is in deadlock would be beneficial. This would be based upon dynamically interrogating a process/resource (tuple) graph to determine which process (if any) is involved in deadlock.

The last goal of the long term plan is to continue to provide better tools for the environment as it evolves. Without a proper set of tools, no environment would be complete. These tools help the system designer as well as the Linda programmer in improving overall system performance. For instance, the Linda monitor constructed in the short term plan is altered to feed information back into the system as it is collected. This information is used to intelligently instantiate processes across the system. Processes can

be instantiated on workstations based on the estimated length of execution times as well as relationships with other processes and partitioned tuple spaces. In addition, process migration from one workstation to another is possible through the detection of related processes, thus effectively reducing the amount of communications overhead.

IV. Summary and Final Comments

With the increasing user demand for greater computational computing power, the proposed Linda-LAN is a conceptually attractive configuration that provides

- a true parallel computational environment at an economical price,
- greater computing power than conventional uni-processor workstations,
- wider accessibility to a parallel environment, and
- utilization of idle network cpu cycles.

In addition, the Linda paradigm provides the system with a portable and easy to use operational framework. The Linda language gives the programmer the ability to design, develop and implement parallel programs in a comprehensible and structured manner. By maximizing tuple throughput, providing tools, enhancing the Linda language and improving network communications, the proposed objective of building a complete distributed Linda parallel processing environment on a common local area network that attains significant performance is achievable.

REFERENCES

[BERND89] D. Berndt, "C-Linda reference Manual (DRAFT) Beta Version 2.0," *Scientific Research Associates*, January 1989.

[BJORN88] R. Bjornson, N. Carriero, D. Gelernter and J. Leichter, "Linda, the Portable Parallel," *Research Report YALE/DCS/RR-520*, January 1988.

[BJORN89] R. Bjornson, N. Carriero, and D. Gelernter, "The Implementation and Performance of Hypercube Linda," *Research Report YALEU/DCS/RR-690*, March 1989.

[CARRI87] N. Carriero, "Implementation of Tuple Space Machines," *Research Report YALEU/DCS/RR-567* (PhD thesis), December 1987.

[CARRI88] N. Carriero and D. Gelernter, "Applications Experience with Linda," *Proc. ACM Symp. Parallel Programming*, July 1988.

[CARRI89a] N. Carriero and D. Gelernter, "Coordination Languages and their Significance," *Yale Tech Report,* YALEU/DCS/RR-716, July 1989.

[CARRI89b] N. Carriero and D. Gelernter, "Linda in Context," *Communications of the ACM*, Vol. 32, No. 4, April 1989.

[DAVID89] C. Davidson, "Technical Correspondence on *Linda in Context,*", Communications of the ACM, Vol. 32, No. 10, October 89, pp.1249-1252.

[GELER85a] D. Gelernter, "Generative Communication in Linda," ACM *Transactions on Programming Languages and Systems*, Vol. 7, No. 1, January 1985, Pages 80-112.

[GELER85b] D. Gelernter, N. Carriero, S. Chandran and S. Chang, "Parallel Programming in Linda," *Proceedings of the 1985 International Conference on Parallel Processing*, August 1985, pp.255-263.

[GELER90] D. Gelernter, "Ada-Linda: Motivation, Informal Description and Examples," *Yale Tech Report.*

[KAHN89] K. Kahn and M. Miller, "Technical Correspondence on *Linda in Context, Communications of the ACM*, Vol. 32, No. 10, October 1989, pp. 1252-1253.

[LELER88] Wm. Leler, "PIX, the latest NeWS," *Cogent Technical Report,* Cogent Research, November 1988.

[LELER90] Wm. Leler, "Linda Meets Unix," *IEEE Computer*, February 1990, pp.43 - 54.

[WHITE88] R. Whiteside and J. Leichter, "Using Linda for Supercomputing On a Local Area Network," in *Proc. Supercomputing '88*, November 1988.

[ZENIT90] S. E. Zenith, "Linda Coordination Language; subsystem kernel architecture (on transputers)," *Research Report YALEU/DCS/RR-794*, May 1990.