Linda (coordination language)

From Wikipedia, the free encyclopedia

In computer science, **Linda**[1] is a model of coordination and communication among several parallel processes operating upon objects stored in and retrieved from shared, virtual, associative memory. Linda was developed by David Gelernter and Nicholas Carriero at Yale University and is named for Linda Lovelace, an actress in the porn movie Deep Throat, a pun on Ada's tribute to Ada Lovelace.[2]

Contents

Model

This model is implemented as a "coordination language" in which several primitives operating on ordered sequence of typed data objects, "tuples," are added to a sequential language, such as C, and a logically global associative memory, called a tuplespace, in which processes store and retrieve tuples.

The original Linda model requires four operations that individual workers perform on the tuples and the tuplespace:

- **in** atomically reads and removes—consumes—a tuple from tuplespace
- **rd** non-destructively reads a tuplespace
- **out** produces a tuple, writing it into tuplespace
- **eval** creates new processes to evaluate tuples, writing the result into tuplespace

Evaluation

Compared to other parallel-processing models, Linda is more orthogonal in treating process coordination as a separate activity from computation, and it is more general in being able to subsume various levels of concurrency —uniprocessor, multi-threaded multiprocessor, or networked—under a single model. Its orthogonality allows processes computing in different languages and platforms to interoperate using the same primitives. Its generality allows a multi-threaded Linda system to be distributed across multiple computers without change.

Whereas message-passing models require tightly-coupled processes sending messages to each other in some sequence or protocol, Linda processes are decoupled from other processes, communicating only through the tuplespace; a process need have no notion of other processes except for the kinds of tuples consumed or produced (data coupling).

Researchers have proposed more primitives to support different types of communication and co-ordination between (open distributed) computer systems, and to solve particular problems arising from various uses of the model. Researchers have also experimented with various means of implementing the virtual shared memory for this model. Many of these researchers proposed larger modifications to the original Linda model, developing a family of systems known as Linda-like systems and implemented as orthogonal technology (unlike original version). An example of this is the language Ease designed by Steven Ericsson-Zenith.

Implementations

Linda was originally implemented in C and Fortran, but has since been implemented in many programming languages, including:

- C: C-Linda, TCP-Linda (http://lindaspaces.com/products/linda.html), LinuxTuples (http://linuxtuples.sourceforge.net/)
- C++: CppLinda (http://sourceforge.net/projects/cpplinda/)
- Erlang: Erlinda (http://code.google.com/p/erlinda/)
- Java: JavaSpaces, TSpaces (http://www.almaden.ibm.com/cs/TSpaces/), LightTS (http://lights.sourceforge.net/), LIME (http://lime.sourceforge.net/index.html)
- Lisp
- Lua: LuaTS (http://www.jucs.org/jucs_9_8/luats_a_reactive_event/Leal_M_A.pdf) Lua Lanes (http://kotisivu.dnainternet.net/askok/bin/lanes/)
- Prolog: SICSus Prolog Linda (http://www.sics.se/sicstus/docs/3.7.1/html/sicstus_29.html)
- Python: PyLinda
- Ruby: Rinda

Some of the more notable Linda implementations include:

- C-Linda or TCP-Linda - the earliest commercial and a widespread implementation of virtual shared memory for supercomputers and clustered systems from Scientific Computing Associates, founded by Martin Schultz.
- JavaSpaces - a Java-based tuplespace implementation that helped popularize distributed computing.
- TSpaces (http://www.almaden.ibm.com/cs/TSpaces/) - a Java-based tuplespace platform from IBM.

Criticisms

Criticisms of Linda from the multiprocessing community tend to focus on the decreased speed of operations in Linda systems as compared to MPI systems.[*citation needed*] While not without justification, these claims were largely refuted for an important class of problems.[3] Detailed criticisms of the Linda model can also be found in Steven Ericsson-Zenith's book *Process Interaction Models*.[4]

Publications

- Gelernter, David; Carriero, Nicholas (1992). "Coordination Languages and their Significance". Communications of the ACM. doi:10.1145/129630.129635 (http://dx.doi.org/10.1145%2F129630.129635).
- Carriero, Nicholas; Gelernter, David; Mattson, Timothy; Sherman, Andrew (1994). "The Linda Alternative to Message-Passing systems". Parallel Computing. doi:10.1016/0167-8191(94)90032-9 (http://dx.doi.org/10.1016%2F0167-8191%2894%2990032-9).

- Wells, George. "Coordination Languages: Back to the Future with Linda" (http://wcat05.unex.es/Documents/Wells.pdf) (PDF). Rhodes University.
- Sluga, Thomas Arkadius. "Modern C++ Implementation of the LINDA coordination language". University of Hannover.

See also

- Dataflow
- Data flow diagram
- Dataflow programming
- Flow-based programming
- Programming in the large and programming in the small

References

1. ^ Ahuja, Sudhir (AT&T Bell Laboratories); Carriero, Nicholas; Gelernter, David (August 1986), "Linda and Friends", *Computer* (IEEE) **19** (8): 26–34
2. ^ Markoff, John (January 19, 1992). "David Gelernter's Romance With Linda" (http://www.nytimes.com/1992/01/19/business/david-gelernter-s-romance-with-linda.html?scp=1&sq=David%20Gelernter%20began%20doctoral%20studies%20Stony%20Brook&st=cse&pagewanted=all). *The New York Times*.
3. ^ Carriero et. al. (1 April 1994). "The Linda Alternative to message-passing systems". *Parallel Computing* **2** (4): 633–655.
4. ^ Ericsson-Zenith (1992). *Process Interaction Models*. Paris University.

External links

- Coordination Language (http://www.jpaulmorrison.com/cgi-bin/wiki.pl?CoordinationLanguage) - A small discussion about the differences between the approach of Linda and that of Flow-based programming
- Linda for C++ (https://sourceforge.net/projects/cpplinda/)
- Linda for C (http://www.comp.nus.edu.sg/~wongwf/linda.html)
- Erlinda (for Erlang) (http://code.google.com/p/erlinda/)
- Linda for Java (http://www.cs.bris.ac.uk/Publications/Papers/2000380.pdf)
- Linda for Prolog (http://www.sics.se/sicstus/docs/3.7.1/html/sicstus_29.html)
- PyLinda (for Python) (http://code.google.com/p/pylinda/)
- Rinda (for Ruby) (http://segment7.net/projects/ruby/drb/rinda/)
- Linda for Scala (on top of Scala Actors) (http://lucdup.blogspot.com/2009/08/scala-spaces-as-scala-actors.html)
- Linda in a Mobile Environment (LIME) (http://lime.sourceforge.net/) (for nesC)

Categories: Concurrent programming languages | Tree programming languages

---