
Blackboard & Related Models for Parallelism

A Kind of Blackboard

(“Linda”: Prof. David Gelernter, Yale)

- A popular abstraction that can be based on a processor/work pool implementation
- Central repository for work, called a “tuple space” (also T-space, Javaspaces) also functions to communicate argument values and results.
- Procedures to enter and remove work from tuple space, that also synchronize processes.

“Linda” operations

- **out(Tuple)** puts Tuple into tuple space
- **in(Pattern)** removes Tuple matching pattern from tuple space; if no such tuple, waits for one to be present
- **rd(Pattern)** like in(Pattern), but does not remove tuple
- **eval(Expression)** puts Expression in tuple space for evaluation (in parallel); result of evaluation becomes tuple left in tuple space

```
/******  
* File: hello_world.cl  
* Simple C Linda example  
******/
```

```
real_main(int argc, char **argv)  
{  
    int nworker, j, hello();  
    nworker=atoi (argv[1]);  
  
    for (j=0; j nworker; j++)  
        eval ("worker", hello(j));  
    for(j=0; j nworker; j++)  
        in("done");  
  
    printf("Hello_world is finished.\n");  
}
```

Master

```
/** function hello **/  
  
int hello (i)  
int i;  
{  
    printf("Hello world from number %d.\n",i);  
    out("done");  
    return(0);  
}
```

Workers

“Linda”

Advantages/Disadvantages

Linda Implementations

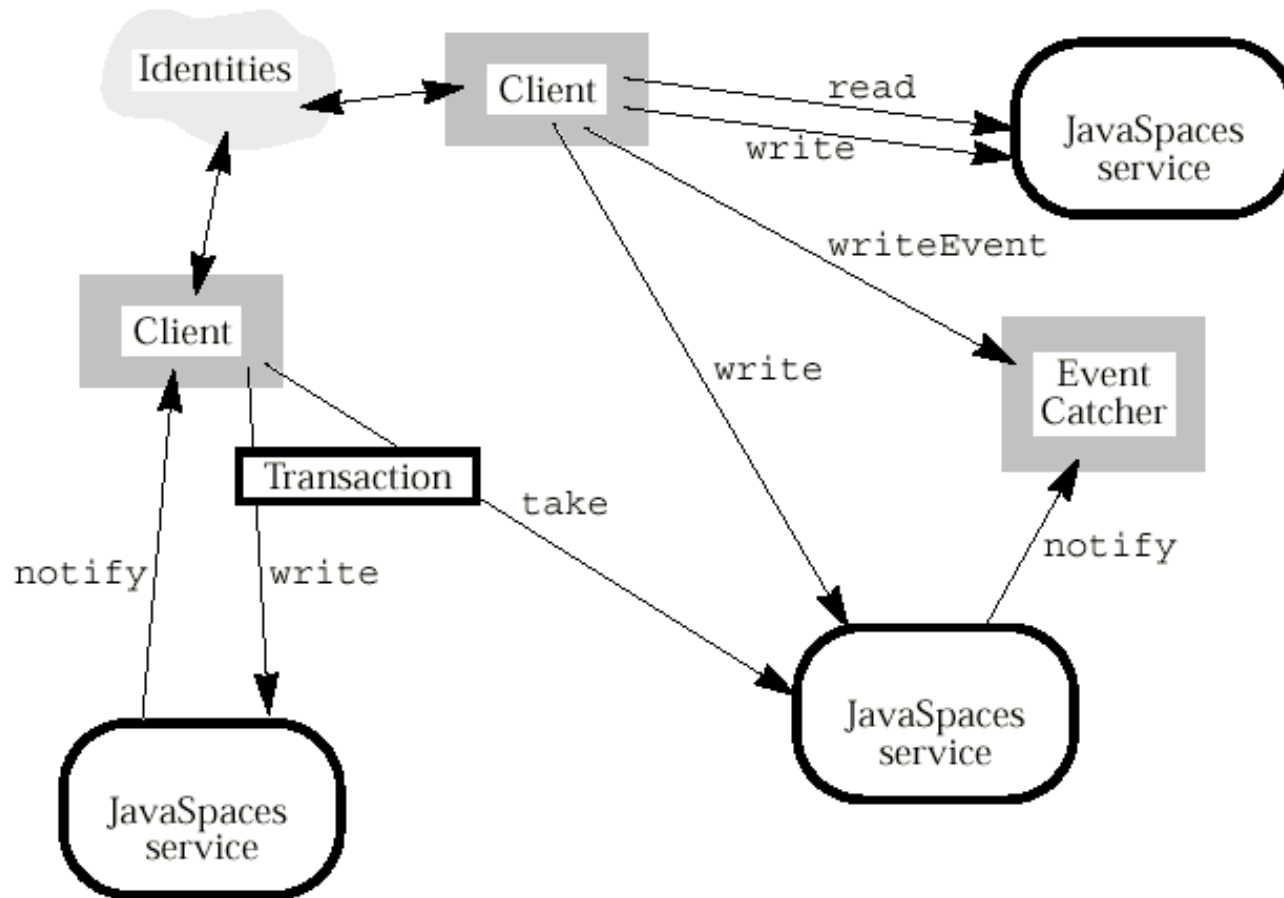
- A former company: [Scientific Computing Associates](#), Inc. founded by Yale profs:
 - C-Linda, Fortran-Linda
 - Pirahna
 - Some version to be in RedHat Linux
- PVM Linda
- C++-Linda
- Polyaspaces (Kevin Eustice, Drew Bernat, HMC '99)

Linda

- How might one implement a Linda atop MPI?

Javaspaces™ and Jini: Linda Derivatives

See <http://developer.java.sun.com/developer/products/jini/>



Jini has moved

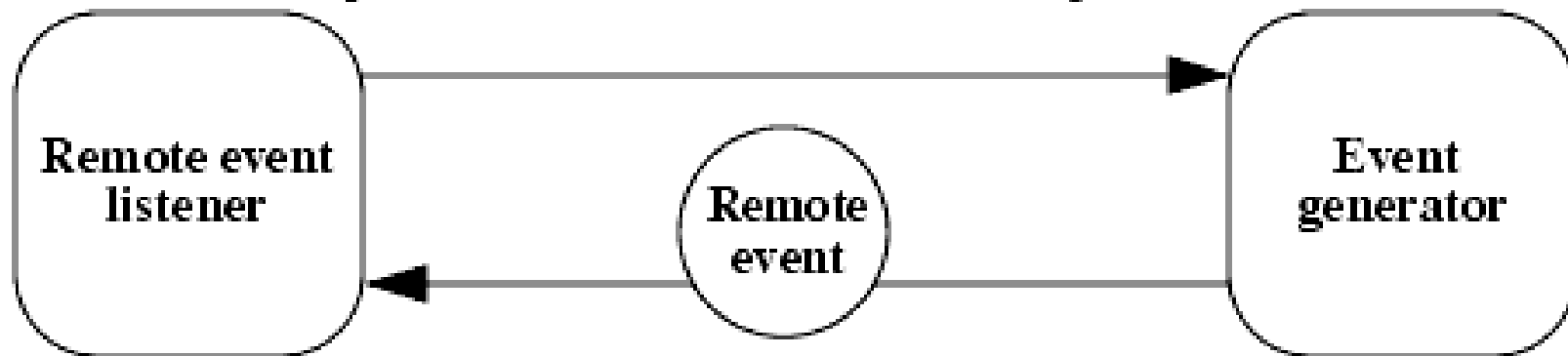
<http://incubator.apache.org/river/RIVER/javadoc.html>

JavaSpaces Interface

- **write**— Write the given entry into this JavaSpaces service.
- **read**— Read an entry from this JavaSpaces service that matches the given template.
- **take**— Read an entry from this JavaSpaces service that matches the given template, removing it from this space.
- **notify**— Notify a specified object when entries that match the given template are written into this JavaSpaces service.
- others, but no **eval**

Jini™ Technology: Remote event listeners

1. The remote event listener registers interest in a particular kind of event with the event generator



2. The event generator fires a remote event to indicate that an event of that kind has occurred

Javaspaces Distributed Example

A book ordering system might look like this:

- A book **buyer** wants to buy 100 copies of a book. He/she writes a **request for bids** into a particular public JavaSpaces service.
- The **broker** runs a server that takes those **requests** out of the space and writes them into a JavaSpaces service for each book seller who registered with the broker for that service.
- A server at each book **seller** removes the **requests** from its JavaSpaces service, presents the request to a human to prepare a bid, and writes the **bid** into the space specified in the book buyer's request for bids.
- When the bidding period closes, the **buyer** takes all the bids from the space and presents them to a human to select the winning bid.

Differences

- Javaspaces is not a remote object service; there is no in-place modification of objects.
- Javaspaces is not a general relational database facility.

A Prolog Implementation of Linda

- Prolog has built-in **pattern-matching** that generalizes that proposed for Linda.
- Prolog can be augmented with a process facility that enables implementing Linda easily.
- The implementation we describe is based on UNIX processes and IPC.
- Thus it is best-suited for coarser grain computations.

Quick Review of Prolog

- Prolog is driven by **goals**, predicate expressions that may contain unbound variables.
- Prolog tries to **solve** a goal by a search process that ends up **binding** the variables, provided that the goal is solvable.
- The rules for solving are logical implications.

Prolog with Multiprocessing (Quintus mp library)

- `mp_fork(+Goal, -Handle)`
+ and - are annotations for input and output
- `mp_create_channel(-Channel)`
- `mp_send(+Channel, +Term)`
- `mp_receive(+Channel, -Term)`
- `mp_select(+List, -Channel, -Term)`

Linda Implementation (1)

init_linda :-

```
mp_create_channel(Channel),  
assert(request_channel(Channel)),  
mp_fork(linda_serve).
```

out(Term) :-

```
request_channel(Channel),  
mp_send(Channel, out(Term)).
```

in(Term) :-

```
result_channel(ResultChannel),  
request_channel(Channel),  
mp_non_blocking_send(Channel, in(Term,ResultChannel)),  
mp_receive(ResultChannel, Term).
```

Linda Implementation (2)

```
linda_serve :-  
    request_channel(Channel),  
    repeat,  
        mp_receive(Channel, Request),  
        process(Request),  
    fail.
```

... about a page of code

Linda with a Specific number of Workers

- See `linda_workers.pl`

```
prolog:1> linda_workers
| ?- ensure_loaded(linda_workers_primes).
| ?- test_primes(1000, 4).
997 991 983 977 971 967 953 947 941 937 929 919 911 907 887 883 881 877
863 859 857 853 839 829 827 823 821 811 809 797 787 773 769 761 757 751
....
281 277 271 269 263 257 251 241 239 233 229 227 223 211 199 197 193 191
181 179 173 167 163 157 151 149 139 137 131 127 113 109 107 103 101 97
89 83 79 73 71 67 61 59 53 47 43 41 37 31 29 23 19 17 13 11 7 5 3
^Z Suspended
prolog:2> ps
  PID TTY          TIME CMD
 14123 pts/30    0:00 prolog
 14124 pts/30    0:00 prolog
 14125 pts/30    0:00 prolog
 14126 pts/30    0:00 prolog
 14127 pts/30    0:02 prolog
 14029 pts/30    0:01 prolog
```

See also

- binProlog:
<http://www.binnetcorp.com/>
Has a Linda, also java, CGI, etc.
- PyLinda

Conclusions on Linda & Friends

- Clearly convenient
- Performance is uncertain

Charm++

