Continuing Thread:
Various Language Approaches

- So far have seen:
  - Posix threads (C, C++)
  - MPI (C, Fortran)
  - PVM (C, Fortran)

- Now:
  - Linda (various)
  - JavaSpaces (Jini)

related to Work Pool model, p 92

“Linda” model
(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, Javaspace) also functions to communicate argument values and results.
- Procedures to enter and remove work from tuple space, that also synchronize processes.
- Also similar to “Blackboard Model” in AI.

“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

“Linda” operations

```c
real_main(int argc , char **argv)
{
   int worker; hello();
   for (j=0; j < worker; j++ )
      eval("worker", hello(j));
   for(j=0; j < worker; j++ )
      in("done");
   printf("Hello world is finished!");
}
/** function hello **/
int hello (i)
int i;
{
   printf("Hello world from number %d",i);
   out("done");
   return(0);
}
```

“Linda”
Advantages/Disadvantages

- more examples:
  - http://www.qpsf.edu.au/software/doc/Linda/cl-examples/
- tutorial: http://www.sca.com/ltutorial.html
**Linda Implementations**

  - C-Linda, Fortran-Linda
  - Pirahna
  - Some version to be in RedHat Linux
  - PVM Linda
  - C++-Linda
  - PolyaSpaces (Kevin Eustice, Drew Bernat)

**Linda**

- Think about how you might implement a Linda atop MPI.

**Javaspaces™: A Linda Derivative**


**Javaspaces Interface**

- **write**—Write the given entry into this JavaSpaces service (like original out).
- **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 (like original in).
- **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 a listener on 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** takes 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.
Book seller example

Differences with Other Technology

- Javaspaces is not a remote object service (like CORBA); 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 interpretable as 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_serve :-
    request_channel(Channel),
    repeat,
    mp_receive(Channel, Request),
    process(Request),
    fail.

... about a page of stuff; see turing:/cs/cs156/prolog/

See also

- binProlog:
  http://www.binnetcorp.com/

- Has a Linda, also java, CGI, etc.

Conclusions on Linda & Friends

- Clearly convenient
- Performance is uncertain

Parallel Mapping in rex

map(F, [x0, x1, x2, x3, ....]) =>

[F(x0), F(x1), F(x2), F(x3), ....]

evaluated concurrently in different threads

There is also a set of primitives in rex that facilitate
Linda implementation Linda construction in rex.