Dana Petcu

Computer Science Department,
Western University of Timisoara
IeAT Research Institute, Timisoara

petcu@info.uvt.ro, http://web.info.uvt.ro/~petcu
Overview

- Motivation
- State-of-the-art
- Architecture and functionality
- Benchmark
- P system simulations
- Conclusions & further work
Motivation

- Suites of benchmarks (e.g. OPS5 2003), shows that rule-based systems need hours to give the solutions when the number of rules to be fired are of thousand order

- Possible solution: parallel, cluster, grid or distributed version of those systems.
State-of-the-art (I)

- Interest in parallel rule-based (production) systems has been raised after 1984
- A comprehensive synthesis of the efforts made before 1994 in providing parallel firing systems in:
  

- Most representative contributions: PESA, Rubic, PARULEL, PPL
- Several parallel and distributed implementations of CLIPS: PCLIPS, DYNACLIPS, CLIPS/hypercubes or PVM-environment
- Above mentioned versions are no more available in general or in the public domain
State-of-the-art (II)

Jess has:
- active development and support,
- elements not present in other production systems
- powerful enough to generate full applications entirely within the Jess system
- core Jess language is still compatible with CLIPS
- a powerful Java scripting environment


State-of-the-art (III)

- Problem: partitioning work among processors in a parallel machine.
- Partition algorithms - common goal:
  - minimize the duplication of working elements,
  - reducing the traffic between processors,
  - balancing the amount of processing in each processor.
- Approaches:
  - **parallel matching approach** parallelize only the match phase -> leads to a limited speedup by the sequential execution of rule.
  - **multiple-rule firing approach** parallelize the match phase and the act phase by firing multiple rules in parallel -> is more promising, but supplementary costs are due to synchronization needs.
  - **special techniques** like copy-and-constraints, compatible rules, analysis of data dependency graph -> increase the parallelism, but do not exploit the parallelism specific to the application domains
  - **task-level parallelism** approach based on the functional decomposition of the problem into a hierarchy of tasks -> leads to better results that the above mentioned ones, but ad-hoc techniques
Parallel Jess – architecture (I)

- parallel version of Jess:
  - distributed memory version
  - based on task parallelism
  - master-workers style of programming
  - use a wrapper

- target architecture: cluster of workstations
Parallel Jess – architecture (II)

Jess wrapper:

1. **Connector** (Java)
   between Jess instance and the system
2. **Messenger** (Java + PVM)
   for communication in the system

Types of Jess instances:

a) **Master**: user controlled via Jess interface
b) **Workers**: remotely controlled, Java embedded version
## Parallel Jess – functionality (1)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(connection)</td>
<td>Establish the socket connection between the Jess instance and the Messenger</td>
</tr>
<tr>
<td>(kernels n)</td>
<td>Launch n embedded Jess instances</td>
</tr>
<tr>
<td>(kill n)</td>
<td>Stop the n embedded Jess instance</td>
</tr>
<tr>
<td>(send n t s)</td>
<td>Send to the nth Jess instance the message labelled t and containing the string s; if n is -1, the message is broadcast to all Jess instances; the function call is a non-blocking operation</td>
</tr>
<tr>
<td>(recv n t)</td>
<td>Returns a string, the message labelled t and received from the nth Jess kernel; if n is -1, the first message from any kernel is accepted; if t is -1, it is any label is accepted; if no message -&gt; the execution is blocked</td>
</tr>
<tr>
<td>(prob n t)</td>
<td>Tests if a message labelled t send by the nth instance has arrive; returns ”t&quot; or “f&quot;; the meaning of -1 is the same as in the case of recv; the function call is a non-blocking operation</td>
</tr>
<tr>
<td>(stop)</td>
<td>Stop all embedded Jess instances and close the socket connection with the Connector and the Messenger</td>
</tr>
</tbody>
</table>
Parallel Jess – functionality (II)

Jess> (batch "ParJess.clp")
   TRUE
Jess> (connection)
   Connection established
Jess> (kernels 2)
   2 kernels launched
Jess> (send -1 1 "(* ?*p* (pi))")
   Multicast successful
Jess> (recv 1 2)
   "3.141592653589793"
Jess> (recv 2 2)
   "6.283185307179586"
Jess> (stop)
   Connection closed
Jess> (exit)
First test (I)

- Use a classical benchmark from OPS5-2003 (www.pst.com/benchcr2.htm)
- Miss Manners:
  - finding an acceptable seating arrangement for guests at a dinner party, by attempting to match people with the same hobbies, and to seat everyone next to a member of the opposite sex.
  - classical solution employs a depth-first search approach to the problem
  - variables of the problem:
    guests no., chairs no., max&min hobbies no.
First test (II)

- data are generated randomly, guests no. of opposite sex are equal
- easier case: e.g. max=3 hobbies, min=2 hobbies
  depth-first search is building a solution relative fast;
  solving time increases exponentially with no. of guests
  e.g. PIV at 2.2 GHz with 512 Mb RAM,
  64 guests: 7 s, 128 guests: 110 s, 256 guests: 1801 s
- more complicated: e.g. min=1 hobby
  depth-first search explores several branches of the search tree until it reach to a solution.
  e.g. max = 2 hobbies,
  64 guests: 2103 s, 128 guests: 7361 s.
First test (III)

- Construct an algorithm to apply task parallelism:
  - the data set is split into $p$ equal fragments for $p$ processors
  - same rules are applied for the depth-first search on each task, but on the different data fragment
  - pre-processing phase

- Cluster environment: 8 IBM PCs at 1.5 GHz and 256 Mb RAM connected by a Myrinet switch at 2Gb/s.
First test (IV)

Using Parallel Jess for Miss Manners problem
256 guests, max 3 hobbies, min 2 hobbies,
only one computer

<table>
<thead>
<tr>
<th>No. instances</th>
<th>Running time</th>
<th>No. of fired rules/instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2321 seconds</td>
<td>33406 rules</td>
</tr>
<tr>
<td>2</td>
<td>214 seconds</td>
<td>8510 rules</td>
</tr>
<tr>
<td>4</td>
<td>35 seconds</td>
<td>2206 rules</td>
</tr>
<tr>
<td>8</td>
<td>11 seconds</td>
<td>590 rules</td>
</tr>
</tbody>
</table>
First test (V)

Running time improvement (and speedup) by using several nodes of the cluster

<table>
<thead>
<tr>
<th>No. nodes</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. instances</td>
<td>1</td>
<td>2321s</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>214s</td>
<td>112s (1.91)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>35s</td>
<td>19s (1.84)</td>
<td>11s (3.18)</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>11s</td>
<td>6s (1.83)</td>
<td>4s (2.75)</td>
<td>3s (3.67)</td>
</tr>
</tbody>
</table>
natural computing, model inspired from the structure and the *functioning of living cells*

express the way *biological substances are modified or moved* among compartments

each compartment delimited from the rest by a membrane is a *computing unit* with own data and its local program (reactions).

all compartments considered as a whole (the cell) are seen as *unconventional computing device* characterized by a membrane structure, where membranes can be hierarchically placed inside a unique external membrane delimiting the entire cell.

all membranes are *semi-permeable barriers*: substances move in or out and change their location in the membrane structure; movement of some substances can be locked.
P system – how is formalized?

- a membrane structure: described by a finite string of matching parentheses
- substances and reactions are represented by objects and evolution rules
- objects are described as symbols or strings over a given alphabet
- evolution rules are given as rewriting rules
- the rules act on objects, by modifying and moving them, and they can also affect the membrane structure, by dissolving or dividing the membranes.
- computation in P systems: starting from an initial configuration (objects and the rules), let the system to evolve.
- application of rules in a non-deterministic and maximal parallel manner
- computation halts when no rule can be further applied
- output: objects sent out to the external membrane or collected inside a specified membrane.
P system - formalism

P-system: \((\Sigma, H, \mu, \omega_1, \ldots, \omega_N, R)\)

\(N\): the number of initial membranes (system degree)
\(\Sigma\): the alphabet of symbol-objects
\(H\): finite set of membrane labels
\(\mu\): the membrane structure
\(\omega_1, \ldots, \omega_N\): strings over \(\Sigma\), the initial multisets of objects, placed in each membrane of \(\mu\)
\(R\): a finite set of evolution rules of the following forms:
- object evolution rules (inside evolution): \([a \rightarrow b]^p_h\)
- send-in rules (incoming objects): \(a[]^{p_1}_h \rightarrow [b]^{p_2}_h\)
- send-out rules (outgoing objects): \([a]^{p_1}_h \rightarrow b[]^{p_2}_h\)
- dissolution rules (membrane dissolved): \([a]^p_h \rightarrow b\)
- division rules (membrane multiplication): \([a]^{p_1}_h \rightarrow [b]^{p_2}_h [c]^{p_3}_h\)

where \(a, b, c \in \Sigma, \ h \in H, \ p, p_1, p_2, p_3 \in \{+, -, 0\}\)
Validation problem

*Given a boolean formula in conjunctive normal form, to determine whether or not is a tautology.*

Consider the problem input in the form

\[ \wedge_{i=1}^{m} \vee_{j=1}^{k_{i}} x_{ij} \] where \( x_{ij} \in \{ X_1, \ldots, X_n, \bar{X}_1, \ldots, \bar{X}_n \} \)

the P system solves the *NP-complete problem* in 
5n+2m+4 evolution steps 
(in the external membrane it is obtained `Yes' or `No').

The number of membranes (the P system degree) 
increases by division from only 3 initially  
(two internal ones, plus the external one) to  
\( 2^{n+2} \) at the computation end.
P system for validation problem

\[ N = 2 \]
\[ \Sigma = \{ x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n \} \cup \{ c_k : 1 \leq k \leq m+1 \} \]
\[ \cup \{ d_k : 1 \leq k \leq 2n + 2m + 2 \} \cup \{ e_k : 0 \leq k \leq 3 \} \]
\[ \cup \{ r_{ik} : 0 \leq i \leq m, 1 \leq k \leq 2n \} \cup \{ \text{Yes, No} \} \]

\[ H = \{1, 2\} \]
\[ \mu = \{[a]_1, [b]_1\} \]
\[ \omega_1 = \{e_1\}, \omega_2 = \{d_1, \text{<the input symbols>}\} \]

\[ R = \{ [d_k]_2^0 \rightarrow [d_k]_2^+ [d_k]_2^- : 1 \leq k \leq n \} \cup \]
\[ \{ [x_{i1} \rightarrow r_{i1}]_2^+ , [\overline{x}_{i1} \rightarrow r_{i1}]_2^-, [x_{i1} \rightarrow \overline{x}_{i1}]_2^-, [\overline{x}_{i1} \rightarrow \overline{x}_{i1}]_2^{+} : 1 \leq i \leq m \} \]
\[ \{ [x_{ij} \rightarrow x_{i,j-1}]_2^+, [x_{ij} \rightarrow x_{i,j-1}]_2^-, [\overline{x}_{ij} \rightarrow \overline{x}_{i,j-1}]_2^+, [\overline{x}_{ij} \rightarrow \overline{x}_{i,j-1}]_2^- : 1 \leq i \leq m, 2 \leq j \leq n \} \]
\[ \{ [d_k]_2^+ \rightarrow d_k [d_k]_2^0, [d_k]_2^- \rightarrow d_k [d_k]_2^0, : 1 \leq k \leq n \} \]
\[ \{ [d_k]_2^0 \rightarrow [d_k-1]_2^0, : 1 \leq k \leq n-1 \} \]
\[ \{ [r_{ik} \rightarrow r_{i,k+1}]_2^0 : 1 \leq i \leq m, 1 \leq k \leq 2n-1 \} \]
\[ \{ [d_k \rightarrow d_{k+1}]_1^0 : n \leq k \leq 3n-3 \} \]
\[ \{ [d_{3n-2} \rightarrow d_{3n-1}e_0]_1^0, e_0 [e_0]_2^0 \rightarrow [c_1]_2^-, [d_{3n-1} \rightarrow d_{3n}]_2^0 \} \]
\[ \{ [d_k \rightarrow d_{k+1}]_1^0 : 3n \leq k \leq 3n+2m \} \]
\[ \{ [r_{12n}]_2^0 \rightarrow r_{12n}]_2^+ \} \]
\[ \{ [r_{i2n} \rightarrow r_{i-12n}]_2^+ : 1 \leq i \leq m \} \]
\[ \{ [r_{20n}]_2^+ \rightarrow [r_{20n}]_2^- \} \cup \{ [c_k \rightarrow c_{k+1}]_2^+ : 1 \leq k \leq m \} \]
\[ \{ [e_{m+1}]_2^- \rightarrow [e_{m+1}]_2^+, [e_{m+1}]_1^0 \rightarrow e_{m+1} [e_{m+1}]_1^+, \{ d_{3n+2m+1}]_2^+ \rightarrow \}
\[ d_{3n+2m+2}]_2^+, [d_{3n+2m+1}]_2^+ \rightarrow [d_{3n+2m+2}]_2^+, \}
\[ [d_{3n+2m+2}]_2^- \rightarrow d_{3n+2m+2]}_2^-, [d_{3n+2m+2}]_2^+ \}
\[ \rightarrow d_{3n+2m+2}]_1^-, [d_{3n+2m+2}]_1^+ \rightarrow d_{3n+2m+2}]_1^- \}
\[ \{ [e_k \rightarrow e_{k+1}]_1^+ : 1 \leq k \leq 2 \} \cup \]
\[ \{ [e_2]_1^+ \rightarrow Yes]_1^+, [e_3]_1^- \rightarrow No]_1^+, [e_1]_1^- \rightarrow No]_1^- \} \]
## Test problem dimension/
Simulation time: Jess on 1 machine

<table>
<thead>
<tr>
<th>$m \times n$</th>
<th>Membranes Rules fired after the last division</th>
<th>Time for firing those rules</th>
<th>Reaching the full membrane configuration</th>
<th>Total time of simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x 2</td>
<td>6</td>
<td>1 s</td>
<td>1 s</td>
<td>2 s</td>
</tr>
<tr>
<td>3 x 3</td>
<td>10</td>
<td>13 s</td>
<td>3 s</td>
<td>16 s</td>
</tr>
<tr>
<td>4 x 4</td>
<td>18</td>
<td>428</td>
<td>6 s</td>
<td>434 s</td>
</tr>
<tr>
<td>5 x 5</td>
<td>34</td>
<td>Out of memory</td>
<td>49 s</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Membranes</th>
<th>1 instance</th>
<th>2 instances</th>
<th>3 instances</th>
<th>4 instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1 s</td>
<td>1 s</td>
<td>3 s</td>
<td>3 s</td>
</tr>
<tr>
<td>10</td>
<td>13 s</td>
<td>5 s</td>
<td>5 s</td>
<td>10 s</td>
</tr>
<tr>
<td>18</td>
<td>428 s</td>
<td>45 s</td>
<td>25 s</td>
<td>33 s</td>
</tr>
<tr>
<td>34</td>
<td>Mem.out</td>
<td>1054 s</td>
<td>325 s</td>
<td>200 s</td>
</tr>
</tbody>
</table>
## Simulation time, speedup and efficiency
### Jess instances on different cluster’s machines

<table>
<thead>
<tr>
<th>Instances</th>
<th>Machines</th>
<th>Membranes</th>
<th>6</th>
<th>10</th>
<th>18</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1 s</td>
<td>13 s</td>
<td>428 s</td>
<td>Memory out</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 2</td>
<td>1 s 1 s</td>
<td>5 s 3 s</td>
<td>45 s 23 s</td>
<td>1054 s 528 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_2$ $E_2$</td>
<td>1 0.51%</td>
<td>1.7 0.65%</td>
<td>1.9 0.95%</td>
<td>2 0.99%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 3</td>
<td>3 s 3 s</td>
<td>5 s 2 s</td>
<td>25 s 10 s</td>
<td>325 s 126 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_3$ $E_3$</td>
<td>1 0.33%</td>
<td>2.5 0.83%</td>
<td>2.5 0.83%</td>
<td>2.6 0.87%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1 4</td>
<td>3 s 2 s</td>
<td>10 s 3 s</td>
<td>33 s 9 s</td>
<td>200 s 52 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_4$ $E_4$</td>
<td>1.5 0.37%</td>
<td>3.3 0.82%</td>
<td>3.7 0.91%</td>
<td>3.8 0.96%</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions & further work

- Development of Parallel Jess - approach task parallelism + wrapper + master-workers programming style
- Parallel Jess exists as a demonstrator system
- Changing JPVM -> MPI will allow the migration towards grids
- A new version for parallel CLIPS on the same principles is an ongoing-work
- The efficiency of Parallel Jess was demonstrated in also in the context of speedup current simulators for membrane computing
- Further experiments with Parallel Jess are needed to include real applications