Research on Distributed Software at RISC-Linz

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

Research and Development Projects

- Distributed Constraint Solving for Functional Logic Programming
  - CFLP project (AITEC).
  - SCORE/Tsukuba, RISC-Linz.
- Distributed Scientific Visualization
  - CASA: Algebraic Geometry in Maple (Franz Winkler et al).
  - Distributed Maple.
- Computer-Supported Education
  - DAJ: Distributed Algorithms in Java.
  - Embedding of executable contents into education material.
- Computer-Supported Distributed Meetings
  - COOPERATE: Universities of Linz, Salzburg, Vienna.
  - Reference project on telecooperation.

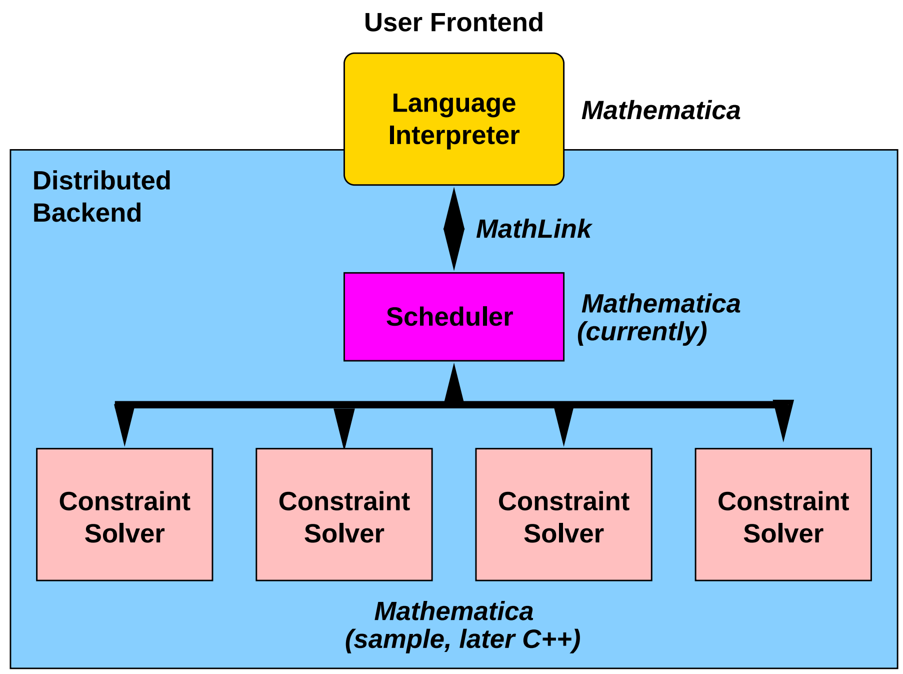
Distributed Constraint Solving for Functional Logic Programming

Mircea Marin (RISC-Linz), Tetsuo Ida (SCORE/Tsukuba).

- Basis: Functional-Logic Programming.
  - Lazy Narrowing Calculus (LNC): Prof. Ida at SCORE/Tsukuba.
  - Implementation of interpreter in *Mathematica*.
- Add: constraint-solving capabilities.
  - Equations/inequalities over special domains (complex, reals).
  - Methods developed in computer algebra.
  - Extend calculus to handle special operators.
- Method: connect many external solvers.
  - Solvers distributed over network.
  - Connected to interpreter via *MathLink*.

CFLP System Architecture

- CFLP language interpreter in Mathematica.
  - Text frontend.
  - Hypertext frontend (*notebooks*).
- External scheduler.
  - Classification of constraints submitted by interpreter.
  - Scheduling constraints among idle constraint solvers.
- Remote solvers.
  - Mathematica solvers.
  - Special solvers in C/C++.



CFLP Language

- LCNC calculus (Hamada et al.) extended by
  - Special inference rules for higher order narrowing (Ida et al.),
  - Partitioning set of function symbols into defined functions (handled by interpreter), constructors (uninterpreted) and external (handled by solvers).
  - Annotations to express (AND-/OR-)parallelism in program goals.
- Currently supported solvers (over complex numbers).
  - `Linear` (linear equations, simplex algorithm),
  - `Polynomial` (polynomial equations, Gröbner bases algorithm),
  - `Derivative` (ordinary differential equations),
  - `Partial` (partial differential equations).

CFLP Example

- System

$$S : \begin{cases} f[x] = g[y] \\ g[y] = g[z^2 - 1] \\ h'[u] = z * u^y \end{cases}$$

$$f[x] \longrightarrow g[y] \Leftarrow x + y = 3 \wedge x^2 - y = 3$$

- Call

`In[2]:=` $S := \{f[x] \approx g[y], g[y] \approx g[z * z - 1], h'[u] \approx z * u^y\};$

`In[3]:=` **TSolve** $\mathbf{[}S, \{x\text{::Compl}, y\text{::Compl}, z\text{::Compl}, h\}, \{u\},$
$\text{Rule->}\{f[x] \longrightarrow g[y] \Leftarrow (x + y \approx 3) \wedge (x^2 - y \approx 3)\},$
$\text{DefinedFunctionSymbol->}\{f\text{::Compl} \to \text{Compl}\},$
$\text{Constructor->}\{g\text{::Compl} \to \text{Compl}\}\mathbf{]}$

- Result

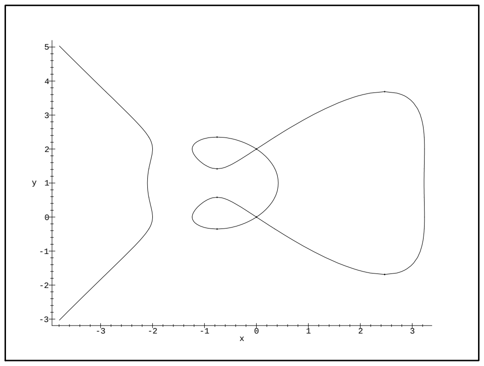
`Out[3]:=` $\{\ \{x \to 2, y \to 1, z \to -\sqrt{2}, h \to \lambda[\{u\}, c1 - \frac{u^2}{\sqrt{2}}]\}, \ldots\}$

Distributed Scientific Visualization

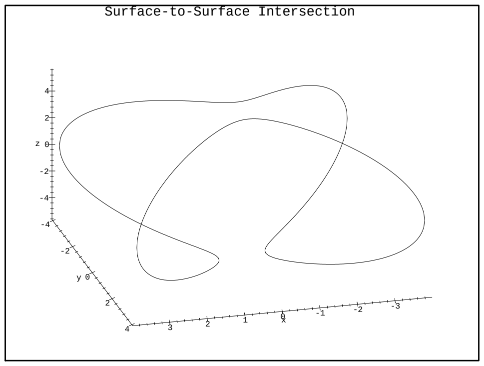
- CASA
  - Computer Algebra Software for Constructive Algebraic Geometry.
  - Package of Maple programs.
  - Franz Winkler et al., 1990–
- Algebraic Geometry.
  - Computations and reasoning about geometric object.
  - Representation as algebraic sets.
  - Manipulation by computer algebra methods.
- Application (e.g.): Scientific Visualization.
  - Plane curves.
  - Surface-to-surface intersections.

Scientific Visualization

$$y^4 - 96a^2y^2 + 100a^2x^2 - x^4$$



$$y^4 - 384y^2 + 400x^2 - x^4 + z^3,$$
$$x^2 + y^2 + z^2 - 30$$



Scientific Visualization

- Numerical methods: fast but unreliable.
  - Qualitatively wrong results possible.
  - E.g: missing or non-existing singularities or other critical points.
- Symbolic methods: reliable but slow.
  - Exponential in degree of polynomial.
  - Up to degree 5 or so.
- Hybrid symbolic-numerical methods: considerable faster.
  - Still exponential complexity.
  - Up to degree 10 or so.

Investigation of high-dimensional curves/surfaces only possible by parallel/distributed processing.

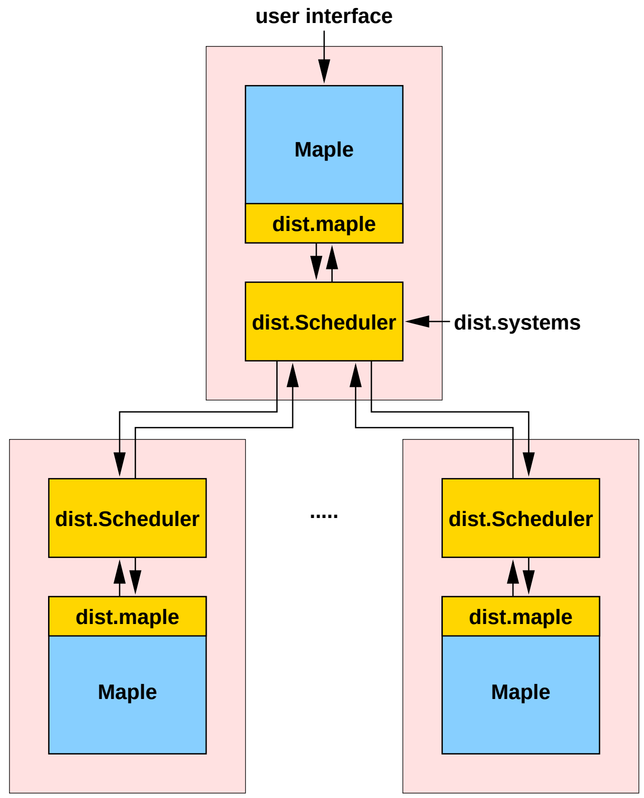
Practical Problems

- Large code base based on Maple.
  - Porting to other system unrealistic.
- Maple is a sequential system.
  - No multi-threading.
  - No distributed processing.
- Maple is a closed system.
  - Commercial distribution.
  - No access to internals of system.
  - No modification of kernel possible.
- Only access: I/O interface.
  - Linearization of internal data structures.
  - Input/output to files/pipes.

Distributed Maple

- Scheduler
  - Node interaction and task scheduling
  - Java program *dist.Scheduler*
- Maple Interface
  - Connection Maple kernel to scheduler
  - Maple package *dist.maple*

No modification of internals of Maple kernel necessary!

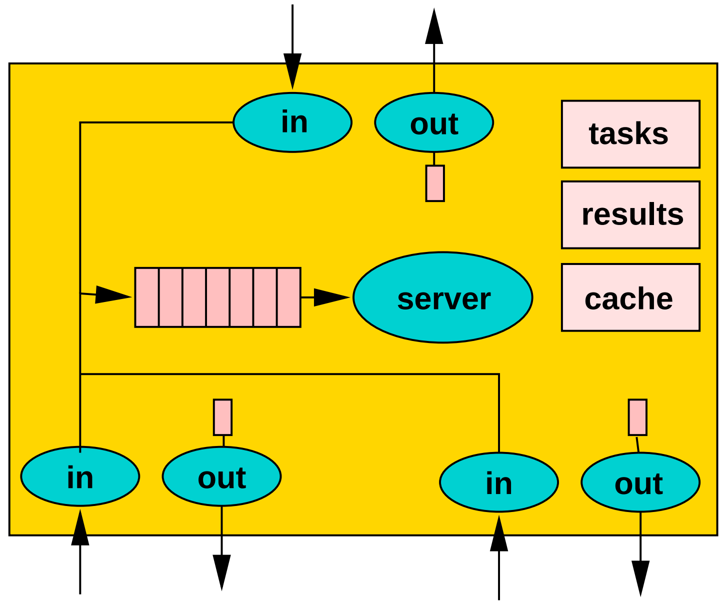


Example

```
    |\^/|     Maple V Release 4 (Johannes Kepler University Linz)
._|\|   |/|_. Copyright (c) 1981-1996 by Waterloo Maple Inc. All rights
 \  MAPLE  /  reserved. Maple and Maple V are registered trademarks of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> read `dist.maple`;
Distributed Maple V1.0 (c) 1998 Wolfgang Schreiner (RISC-Linz)
See http://www.risc.uni-linz.ac.at/software/distmaple
> dist[initialize]([[deneb,solaris], [iris,irix]]);
connecting deneb...
connecting iris...
                                     okay
> t1 := dist[start](int, x^n, x):
> t2 := dist[start](int, x^n, n):
> dist[wait](t1) + dist[wait](t2);
                                (n + 1)     n
                               x           x
                               -------- + -----
                                n + 1     ln(x)
```

Scheduler

- Global Task Pool.
  - Tasks to be scheduled.
  - Task results.
  - Result caches.
- Distributed Operation.
  - Remote system startup.
  - Multi-threaded service handling.
  - Watchdog threads monitor communication.



Simple version of a distributed operating system.

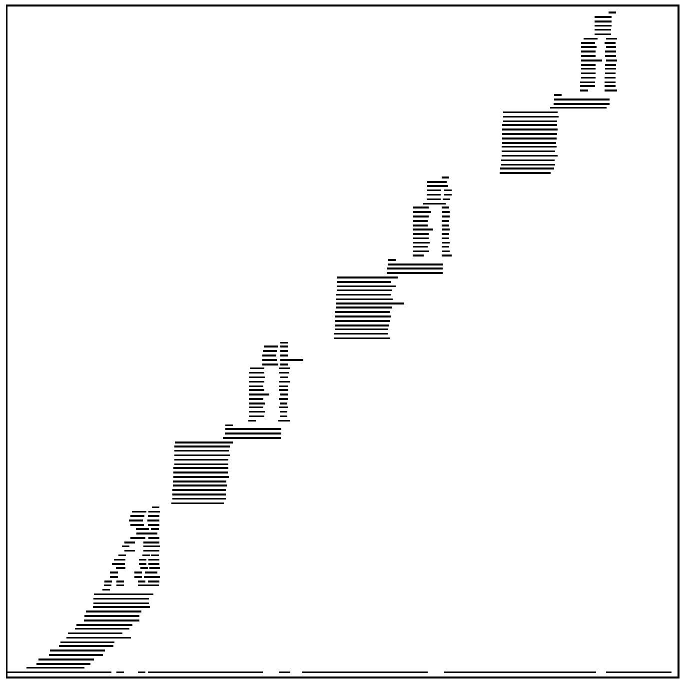
Dynamic Parallelism

```
# list of factorizations of numbers in list
dfactors := proc(numbers)
  ...
  # divide
  m := trunc(length/2);
  numbers1 := [ seq(numbers[i], i=1..m) ];
  numbers2 := [ seq(numbers[i], i=m+1..length) ];

  # conquer
  task1 := dist[start](dfactors, numbers1);
  result2 := dfactors(numbers2);

  # combine
  result1 := dist[wait](task1);
  RETURN ([op(result1), op(result2)]);
end:
```

Recursive parallelization possible!

Distributed Visualization

Parallelization of the core algorithm.



Based on experience with earlier shared-memory parallelization.

Computer-Supported Education

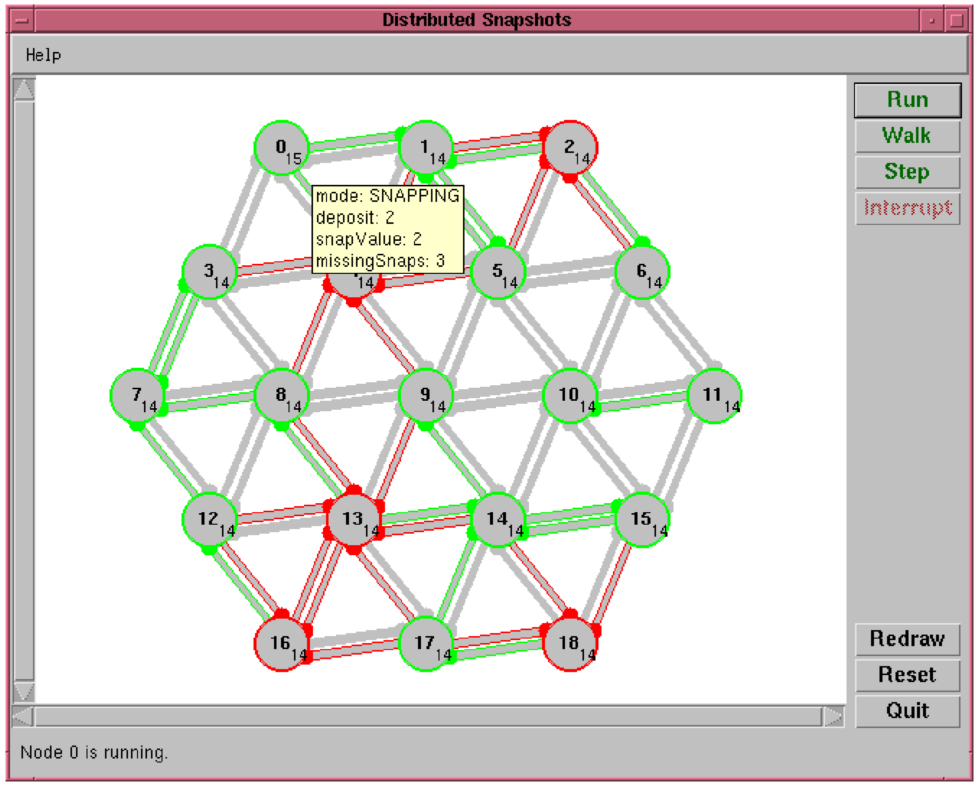
Example: teaching distributed algorithms and programming.

- Problem: hand-on experiments.
  - Paper-and-pencil work too abstract.
  - Real systems (PVM, MPI, . . .) too difficult.
  - No integration of text material with executable programs.
- DAJ: Distributed Algorithms in Java.
  - Java package for writing distributed algorithms.
  - Message-passing based programming model.
  - Standalone execution (with/without visualization).
  - Embedded as applet into an HTML page.

Let students experience the algorithms online.

Executing DAJ Programs

- Running the program.
  - Fast, slow, or step-wise.
  - Interruption, restart, rerun.
- Observing the state of the network.
  - Node ready, blocked, or terminated.
  - Channel empty or holding messages.
  - Contents of node variables and messages.



Writing DAJ Programs

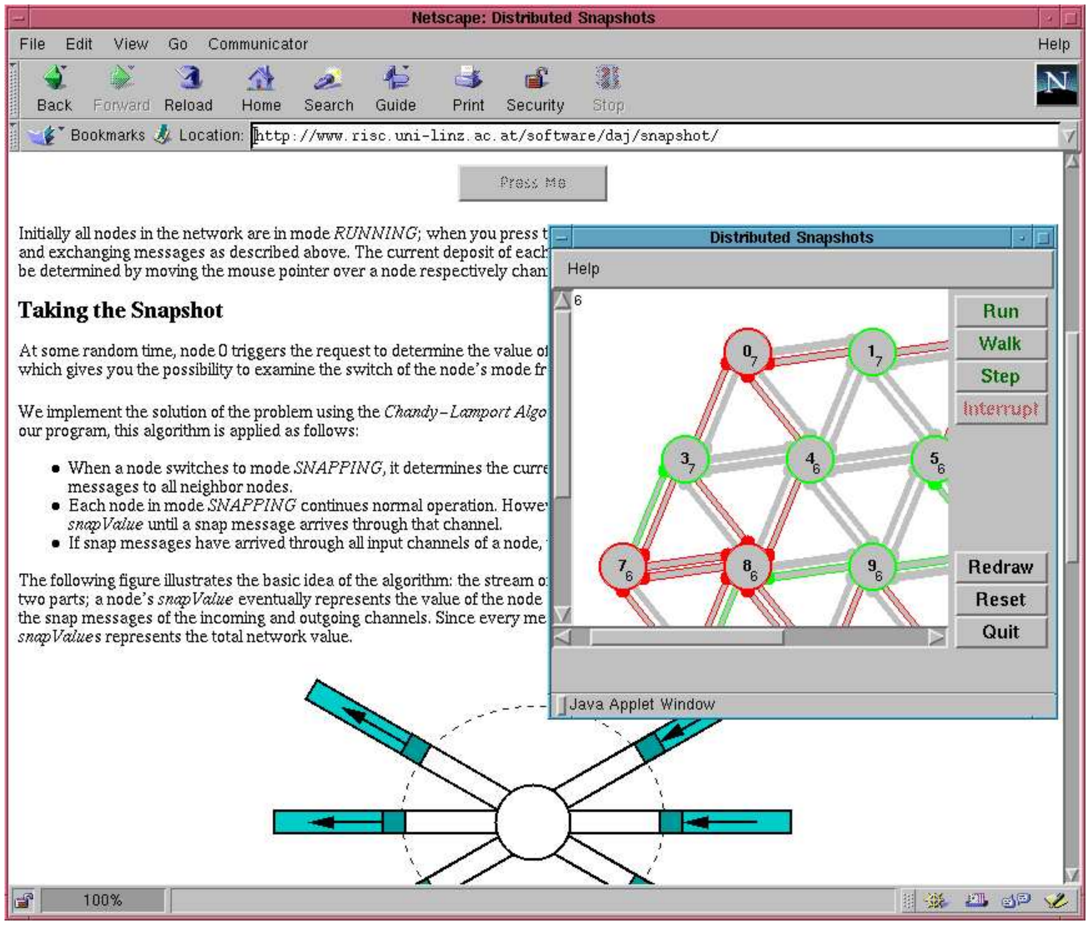
```
import daj.*;
public class Main extends Application
{
  public void construct()
  { Node nodes[][] = new Node[w+h][2*h+1];
    ...
    nodes[i][j] = node(new Prog(size, number, deposit), ...);
    ...
  }
}
class Prog extends Program
{
  public void main()
  { ...
    message = in(index).receive();
    ...
    out().send(resultMessage);
  }
}
```

Checking Global Assertions

```
class NumberOfMessages extends GlobalAssertion
{
  int count, n;
  ...
  public boolean assert(Program prog[])
  {
    count = 0;
    for (int j = 0; j < prog.length; j++)
      {
        Prog program = (Prog)prog[j];
        count += getMessages(program.out(0)).length;
        count += getMessages(program.out(1)).length;
        if (program.msg != null) count++;
      }
    return count == n;
  }
}
```

There are n messages in the network.

Executable Documents



Computer-Supported Distributed Meetings

- Reference project on telecooperation.
  - Supported by Austrian Ministery for Science and Traffic.
  - COOPERATE: Universities of Linz, Salzburg, Vienna.
  - Various software components supporting distributed meetings.
- RISC-Linz: Online generation of meeting records
  - See separate slides.