DelftVM

A new virtual machine for spatial computers

Maurice Bos
Spatial Computing

• “[In a spatial computer,] a program runs on a collection of devices spread through space whose ability to interact is strongly dependent on their geometry.” [1]
  - Sensor networks
  - Swarm robotics
  - …

Spatial Computing

• Examples:
  • Wireless Sensor Network
  • Quadcopter Swarm
  • Interactive Floor

• Scalability
Spatial Computing

- Program the 'spatial computer' as a whole
- Several platforms already available
  - Proto
  - Netlogo
  - Maté
Proto

'The language of space/time.' – J. Beal

• Started as a project at MIT, now at Raytheon BBN Technologies.

• A functional language for spatial computers. [2]
  • Lisp with extra constructs for time and space.
  • Continuous abstraction for space and time.

Virtual Machine

• Why a virtual machine?
  • Viral programming
  • Sand-boxing
  • Portability
Proto VM

- Specific for one language (Proto)
- Execution rounds
- Stack machine
- Primitive data types are floating point values and tuples/vectors
  - A vector acts as just one element on the stack
- Instruction set matchesProto almost 1:1
  - Nice for the compiler, but moves complexity to the virtual machine
DelftVM

• More generic VM for spatial computing
  • Designed for 'spatial computing', not for a specific language

• For small (custom) electronic devices
  • With any form of communication. (wireless, wired, shared memory, anything)

• Mathematic spatial-computing languages
  • Support for floating point and vectors
DelftVM

- Written in C++03, compatible with C++11
  - Only standard freestanding C++
- Platform independent
  - Runs on virtually every platform
- Small and lightweight
  - Uses little ROM and RAM
Architecture

- Primitive types: floats, integers, vectors
  - Dynamic typing
  - Vectors are stored entirely on the stack
  - A 'copy' of a vector is only a reference
Architecture

• Two-stack machine
  • Separate call stack
Instruction Set

• Short instructions
  • One byte per instruction
  • Implicit operands

• Very few instructions
  • Instructions are polymorphic
    • Only one 'add' instruction for integers, floats and even vectors
Instruction Set - Comparison

Compared to Proto:

- Programs are a lot smaller
- Requires less functionality in the VM, but more in the compiler

Example program

DelftVM:
23 bytes
in, 1, cjmp, 14, push inf, skip-nbr, 11, nbr-mem, 0, nbr-range, add, min, next-nbr, -4, jmp, 2, push 0, dup, set-state, 0, jmp, -23

Proto:
60 bytes
def-fun, 2, inf, ret, def-fun, 6, ret, 1, ret, 0, min, ret, def-fun, 14, nbr-range, lit, 0, eq, if, 6, ret, 0, nbr-range, add, jmp, 1, inf, ret, def-fun, 21, glo-ref, 0, init-feedback, 0, let, 1, ret, 0, lit 1, sense, lit 0, glo-ref, 1, glo-ref, 2, ret, 0, fold-hood-plus, 0, mux, feedback, 0, let, 1, ret, 0, pop-let, 2, ret
Communication

• Supports both messages and automatic sharing
  • Instruction to broadcast a range of your memory.
  • (In Proto, your 'public' variables are automatically broadcasted.)
• Instructions to iterate over a list of neighbours.
  • A list of neighbours can be automatically built from the received messages.
Example

Take the maximum of the value of a sensor of all neighbours, and send it to the actuator.

15 bytes

```
in 0
set 1 0
send 1 0
import
lit 0
nbr-mem 1 0
max
next-nbr -5
out 0
jmp -15
```
Future Plans

• A simulator
  • Realistic communication
  • Movement
  • Sensors / Actuators
  • Interface real world
• An assembly-language and an assembler
• A compiler (for Proto and other languages)
Thanks for your attention