Movitz

Frode V. Fjeld

Department of Computer Science
University of Tromsø

June 13, 2004

What is Movitz?

In essence: A platform for OS-level hacking with Common Lisp.

- Exploratory, interactive programming "on the metal" (Poking the parallell port should be *easy*!)
- An integrated run-time and programming environment that is dynamic and introspective from the ground up. (How many problems with software are really due to inter-process confusion?)
- A platform for deploying stand-alone applications, with a minimum of non-essentials. (Does your application *need* interrupts?)

What is Movitz?

Movitz consists of these components:

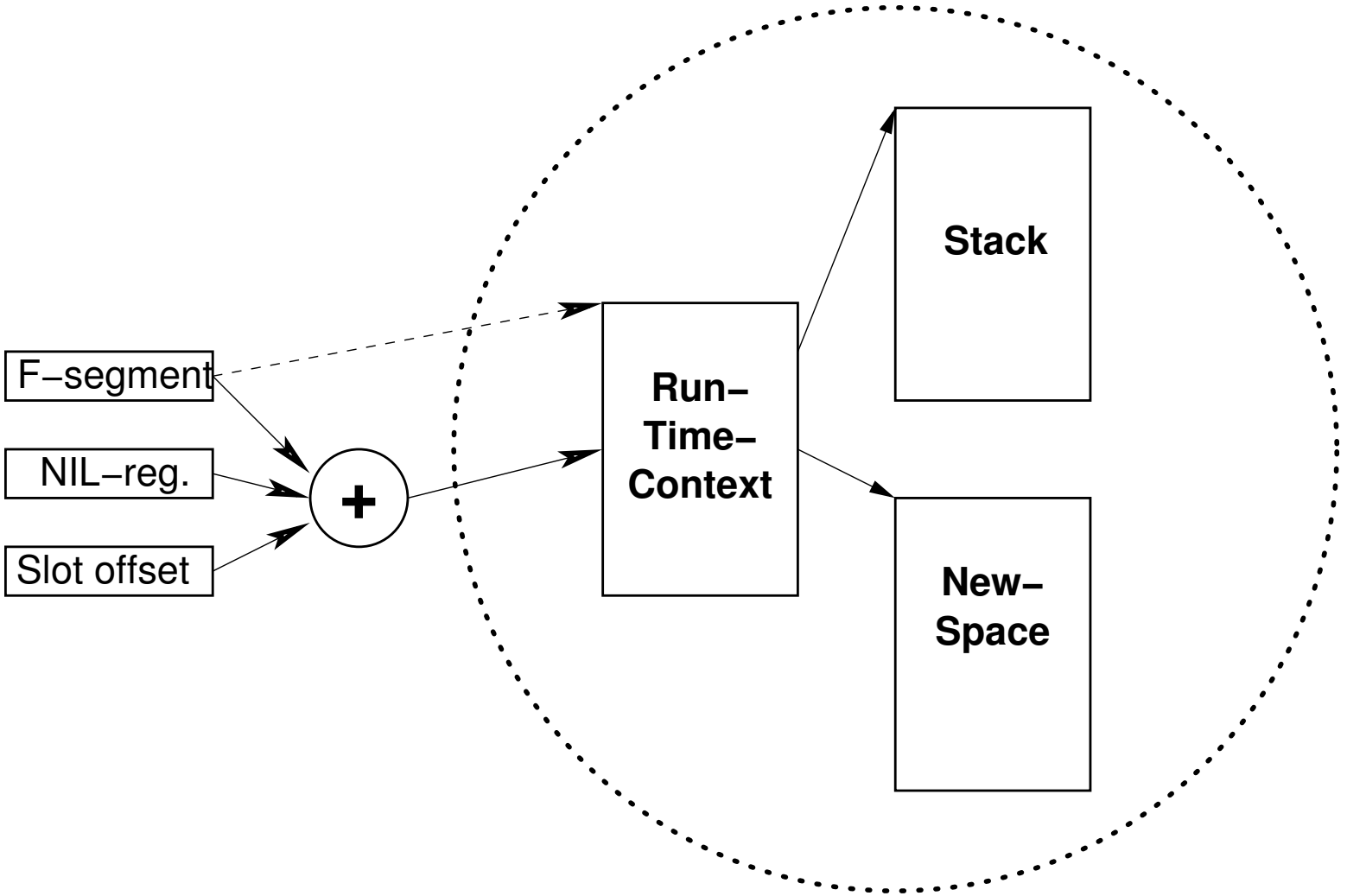
- A minimalistic run-time environment (platform) for Common Lisp programs on x86 PC hardware.
- A compiler that targets this platform.
- Software libraries, including the ANSI CL library, OS drivers and related functionality, etc.

Movitz run-time environment architecture

Attempts to be "minimalistic":

- Provide just the essentials for running CL functions.
- Factor this "core" out from the overall system architecture(s).
- Somewhat like e.g. gcc provides the essentials for running C functions.
- A firm platform that provides for predictable performance, allows experimentation, application/system flexibility/adaptability and specialization.

Movitz run-time environment architecture



Cross-compiler

- The Movitz compiler is a *cross compiler*.
- The *host system* is any ANSI CL implementation.
- The *target system* is the Movitz platform.
- On the host system, a "symbolic image" represents the target lisp world.
- The "symbolic image" can be inspected, modified (i.e. compile individual functions etc.) and dumped to bootable disk images.

Current status

- Compiler produces code comparable in speed to "good" compilers (perhaps 30% overhead wrt. CMUCL on some code).
- Common Lisp features not (yet) supported:
  - Numbers: floating-point, ratios, complexes. (Bignums.)
  - Multi-dimensional and/or adjustable arrays.
  - Streams, pathnames, etc.
  - Target-side interpreted macros (backquote).
  - Read-tables, pretty-printing.
  - Bits and pieces.
- Drivers for keyboard, textmode console, NE2000 NIC, and some minor PC controller chips.

. . . Current status

- Library code, such as readline, IPv4 and IPv6 basics.
- Typical ANSI CL code compiles and runs OK, modulo “big system”-issues and the missing features listed.
- Example kernel that boots up a REPL is about 750 KB.

Future work

- Core run-time environment expected to stabilize when multithreading is in place.
- There's *always* room for improvement in the compiler and CL library.
- Using Movitz as intended: A platform for OS and stand-alone application development and experimentation.

A rather eager Garbage Collector

```
(defun kill-the-newborns ()
  (let* ((oldspace (%run-time-context-slot 'nursery-space))
         (newspace (space-other oldspace)))
    (setf (%run-time-context-slot 'nursery-space) newspace)
    (flet ((zap-oldspace (x location)
             (declare (ignore location))
             (if (object-in-space-p oldspace x)
                 nil
               x)))
      (map-heap-words #'zap-oldspace 0 (malloc-end))
      (map-stack-words #'zap-oldspace (current-stack-frame))
      (initialize-space oldspace)
      (values))))
```

Movitz run-time environment architecture