A Portable Run-Time System for the Hermes Distributed Programming Language

David F. Bacon
Andy Lowry
IBM T.J. Watson Research Center
Contributors to the Hermes System are:

Robert E. Strom
Shaula Alexander Yemini
David F. Bacon
Arthur Goldberg
Andy Lowry
Daniel Yellin
Van L. Nguyen
Joel Auslander
Overview of Presentation

Hermes Philosophy

Hermes Language Overview

Prototype Implementation

Conclusions & Future Work
Hermes Philosophy

Unify Language & System

Simple, High-Level Machine Model

Compiler Should Hide Low-Level Details

Automatic Semantics-Preserving Transformations Improve Performance

Pragmas can give hints to compiler
Hermes Language

Designed for systems that are:

- Large
- Long-Lived
- Dynamic
- Multi-User
- Distributed
Hermes Language

Process Model

- Process is basic unit of modularity
- Role similar to functions in C
- Purely sequential execution
- Process owns local storage & communication ports

→ No shared or global data
→ No pointers
Hermes Language

Hermes is Secure

\[ x := 10; \] \} Does your compiler
\[ \text{print } x; \] \} guarantee that "10" is printed?

Processes interact ONLY by message passing.

\[ \Rightarrow \text{Heavyweight isolation with lightweight efficiency} \]

Security Achieved Through:

- Strong Typing
- Typestate checking
- Minimal Run-time checking

- 99% handled by compiler
Hermes Language

Typestate Checking Detects:

- References to uninitialized variables
- Misuse of variants
- Violations of in-out constraints in interfaces

Example:

```plaintext
x := f(a, b);
if (x < 0) then
    y := 1;
end if;

z := y + 5;
```

↑ possibly uninitialized
⇒ Typestate error
Hermes Language

Communication Ports

- Typed Communication Between Processes
- Synchronous mode replaces procedure call

- Asynchronous mode also available
Hermes Language

Tables

- Homogeneous collections of data
- Replace arrays & pointer-based structures
- Associative access for:
  - Selection
  - Subtable copy
  - Subtable extraction
  - Subtable scan

- Table attributes:
  - ORDERED - elements can be indexed as 0...n
  - KEY - designates portion of each element guaranteed unique within table
Hermes Language

Table Examples

-- Remove leading blanks from string

```lisp
pos := position of c in s where (c <> ' ');
extract junk from c in s where
(position of c < pos);
```

-- Find immediate ancestors of node n of graph G

```lisp
ancestors := every of node in G.nodes where (exists of e in G.edges
where (e.source = node and e.sink = n));
```
**Hermes Prototype**

Compiler turns Hermes source into LI instructions

1. **Parser (C)** produces abstract program object

2. **Type checker (Hermes)**
   - Checks for type errors
   - Infers types of temporaries

3. **Typestate Checker (Hermes)**
   - Checks for typestate errors
   - Generates coercions

4. **Code Generator (Hermes)** produces LI instructions

**Interpreter (C) executes LI instructions**

→ Interpreter is NOT a prerequisite for security

**Run-time system (C) provides access to some Unix services**
Communication Ports

**Local case:**

- Import = message queue
- Output = pointer to import
- Message transfer via pointer manipulation

**Remote case:**

- Import = message queue
- Output = <host id, interpreter id, import address>
- Message transfer over TCP connections using XDR for encoding/decoding data.
  - Sun RPC enhanced for asynchronous, non-blocking communication
  - Socket "cache" avoids most setup & tear-down of channels
Foreign Processes

Mechanism whereby runtime system services are made available

Process PCB contains pointer to "interpreter" function:

- Standard LI interpreter for normal Hermes processes
- Custom C function for foreign process

Interpreter calls PCB's interpreter function when a process is scheduled
Tables

Single High-Level Abstraction Fills Many Needs

- Multiple Implementation Strategies
- Current Interpreter uses:
  - Arrays
  - Linked Lists
  - Character Strings
  - AVL Trees
- Other representations easily added
- Multiple simultaneous representations
- Representations chosen by compiler
Conclusions

- Hermes process model is attractive platform for complex distributed systems
- Typestate Checking is a Big Win!
- Table abstraction is powerful
- Interpretive design makes prototype highly portable
Future Work

More powerful transformations

Interface to other languages

Native code compiler
Get Hermes For Yourself!

* FREE
* No License Agreement
* Complete Sources
* Available throughout the Free World

Current Platforms:
- IBM RISC System/6000 AIX 3.1
- IBM PC/RT 4.3 BSD Unix
- Sun3 & Sun4 SunOS 4.0.3

Requests to:
hermes-request@ibm.com