Why Slow?

A look at the cost of invocations.

Eden Project
Department of Computer Science
University of Washington

Memo: A resume of a meeting held November 18th, 1983.

Participants: Almes, Lazowska, Levy, pete, eric (editor).

Purpose:
A closer look at the cost of Eden invocations. The idea was to pinpoint features of the Eden invocation mechanism which have performance impacts as distinct from identifying *inefficient* details of the current implementation. The list thus excludes things like "the compiler generates poor code" since this is not an inherent feature of epl/invocations.

List. The following is the list of invocation features drawn up at the meeting. [I have added comments in square brackets.]

(1) Operation names are strings.

(2) "Supertypes" are allowed. [That is, an Edentype may fake it and implement the invocation procedures of another Edentype. Right now, I see this feature as something which is only implicitly defined by epl and which just happens to be supported by the current implementation but which is subject to interpretation. I'd like to see more research on what should be supported, or, at least, the semantics of the supported feature should be written up.]

(3) "Not everything is compiled at once." [This problem could require more compiler support.]

(4) Capabilities.
- Location independent names bound to Ejects at invocation time. [Dynamic location procedure must find referenced Eject and prevent multiple active forms from being generated.]
- Ejects can move. [Impacts representation of capabilities and location procedure.]
- Enforcement of access. [Protection of capabilities against forgery is expensive.]

(5) ESCII's.
- Packing and unpacking of parameter lists. [Inherent in any RPC mechanism: Somehow those bytes must be shipped.]
- Type checking. [More compiler support would be helpful.]
- Variable length parameters (parameter lengths bound at invocation time).
- Architecture independence.

(6) Multiple processes within Ejects. [Allows multiple, concurrent invocations.]

(7) Presence of both synchronous and asynchronous invocations.

(8) Binding of UID-to-location (*e.g.*, avoiding multiple activations). [See capabilities above.]

(9) Heterogenous architecture (*e.g.*, flipping bytes in integers.)

(10) Status reporting. [Not on list at the meeting: Invocations return status values which require extra parameters in the call and some work to obtain them.]

(11) Access rights. [Not on list at the meeting: The addition of access right cost some time for setting and checking besides requiring an extra parameter in each invocation and space in capabilities.]

-- eric