



Discussion

Jun Zhang



Why create a new language ?

- Is it better to design a new language with new constructs (abstract types, movable stack frames) or add mobility as an extension to an existing programming language ?
- Do you think the authors were justified in inventing a whole new language, instead of modifying an existing one (a la distributed Smalltalk or uC++)? Do you think that adoption of their solution would have been greater if it had been implemented overtop a contemporary, popular language?



Object-Oriented Properties of Emerald

- Why objects are not class-based and do not form a hierarchy? Why is it costly to separate an object from its code in a distributed environment? What are pros and cons of this instance-based object model?
- Why did emerald leave out other features of a object based system like inheritance and run time dispatching ?



How to handle failure

- Does emerald have any mechanism to deal with node failure ?
- It seems that they don't have any special policy for fault handling. What happens in case of a node failures? Is there anyway to recover object states? What if there is a node with intermittent connectivity? Did they address these issues later? If the answer is no is it practical to use such a system?



Questions on Mobility

- Who sets up the objects initially and what is the criteria for object movement/migration decisions?
- In what situation would you want to move an object?
- About Emerald Processes, objects may move, so what to do about activations?
- Why do we break the process stack when moving an object? We know that we'll likely need that object again anyway so why do we move it?

Questions on Homogeneous System



- The system depends on homogeneous machines ... would it be possible to implement across heterogeneous machine architecture?
- About homogeneity of the system: Even though the system used is homogenous, how possible do you think to use this idea on nonhomogenous systems. (i.e. different architectures in the nodes). One problem that appears is the movement of register values as different architectures have different purpose registers. Can you think of anything else? (i.e. kernel dependencies etc)
- One of the limitations in approach of Emerald is that it works only in homogeneous local area networks. And because the authors didn't include the future work in the paper, do you think which should be developed to overcome that limitation or any thing that need to be improved from the system?



Potential Risk

- One would normally associate a distributed system with attributes such as robustness and reliability. However, does the introduction of mobility actually harm this? (e.g. while the authors assume that nodes can be trusted, what happens if a certain program were to direct all 'important' objects into a single node, and then either maliciously or otherwise, remove the node from the network?)
- Emerald has completely integrated the distribution into the language. Although this approach has several strengths, I think that combining language issues and system issues is not a good idea in general, and this is a weakness for their method. Maybe this is why their method is not in widespread use. What do you think?



Comparison with Related Work

- What does fine-grained mobility offers that is superior or different from RPC?
- Emerald is a system that provides object mobility. How do you compare this system with the mobile-agent paradigm developed recently?
- How does Emerald compare with CORBA? What are the similarities if there are any ?



Later Work or Usage

- This work was done in 1988 .. is there any progeny of this work that is in use today?
- Could we apply similar ideas in explicit data mobility to give a programmer direct control over a cache hierarchy?



Others

- Did emerald use any naming/directory lookup service to locate objects by name ?
- Are the calls in Emerald synchronous or Asynchronous? Why?
- Do you think that keeping the direct memory addressing is the best choice?