Modularity, Reuse and Efficiency with Message-Driven Libraries

L. V. Kale and A. Gursoy

Parallel Programming Laboratory

Department of Computer Science

University of Illinois, Urbana.

The Lure of Reuse

- Parallel Software is harder to develop

  So, bigger benefits if we can reuse it.

- The Challenges for Reuse of Parallel Libraries:

  Context dependences (e.g. data distribution)

  Coordination and mixing of synchronization needs of individual modules.

  Possible loss of efficiency

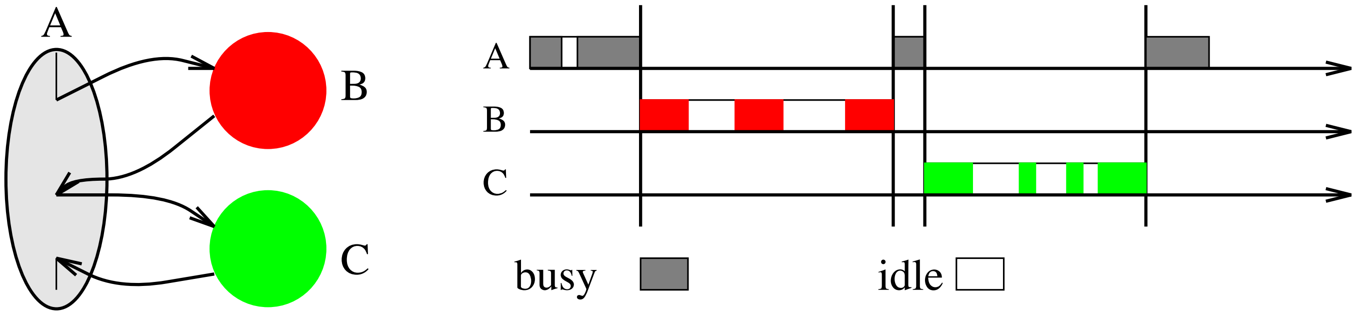
The Requirements for Reuse

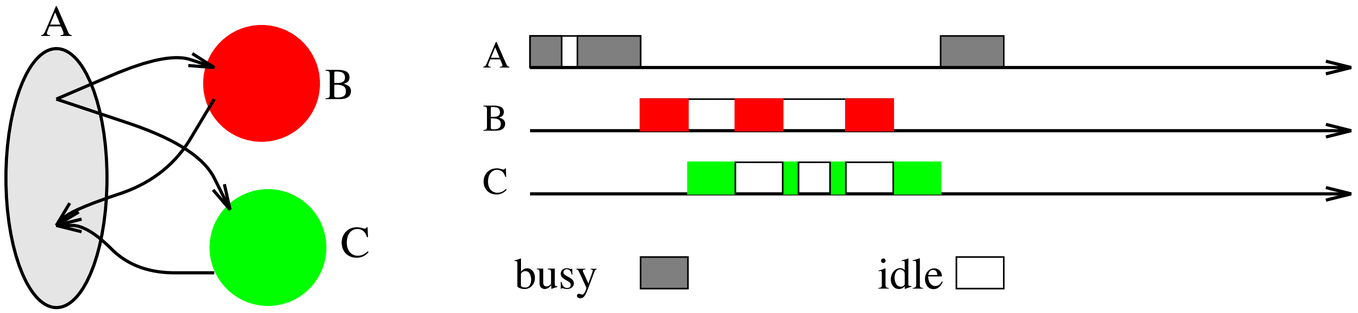
- Modularity and reuse should not entail loss of efficiency.
- Facilitate distributed flow of data across modules.
- Practicality: must permit modules distributed in object format.

Outline

- Message-driven execution
- Branch Office objects
- Static and Dynamic interfaces
- Concurrently reentrant libraries
- Library invocation protocols
- Multilingual interoperability

Message-Driven Execution vs SPMD





Emulating MDE

- Why SPMD can't effectively simulate MDE

Branch office objects

- Global objects with representative on each processor.
- In many applications, two modules may want to distribute data differently.
- Data transfer protocols may become complex, and too specific.
  - e.g. FMA module with Molecular Dynamics
  - *Representatives* provide a universal method for data exchange.

Static and Dynamic Interface

Resolving names/identities

Static: resolved at compile-time

Address name conflicts via module constructs,
explicit export/import.

Dynamic: resolved at run-time

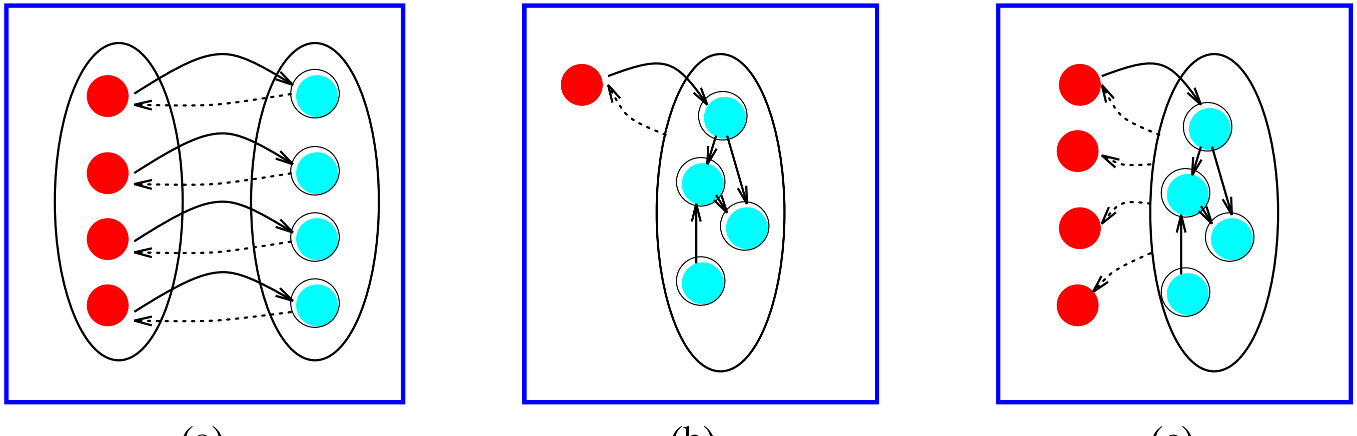
First class object ids,
methods,
functions.

Concurrently ”Reentrant” libraries

- Where needed:
  - Overlapping multiple identical operations
  - Example: concurrent reductions
- How to build:
  - Attach reference numbers to messages and requests
  - Library maintains a separate environment for each reference number

Library Invocation

Protocols for Transfer of data and control across modules.



Multilingual Interoperability

- Many good languages for parallel programming
- Also, libraries being developed in such specialized languages
- Should be able to reuse them across languages
- **Objective:** compose applications by linking Modules written in different languages.
- **Why is this hard:**
  - Languagess may have different scheduling models
  - different ways of dealing with concurrency
  - different control regimes

Concurrency availaibility of alternative actions on a processor at a single point in time: *Allowed or not, how expressed*

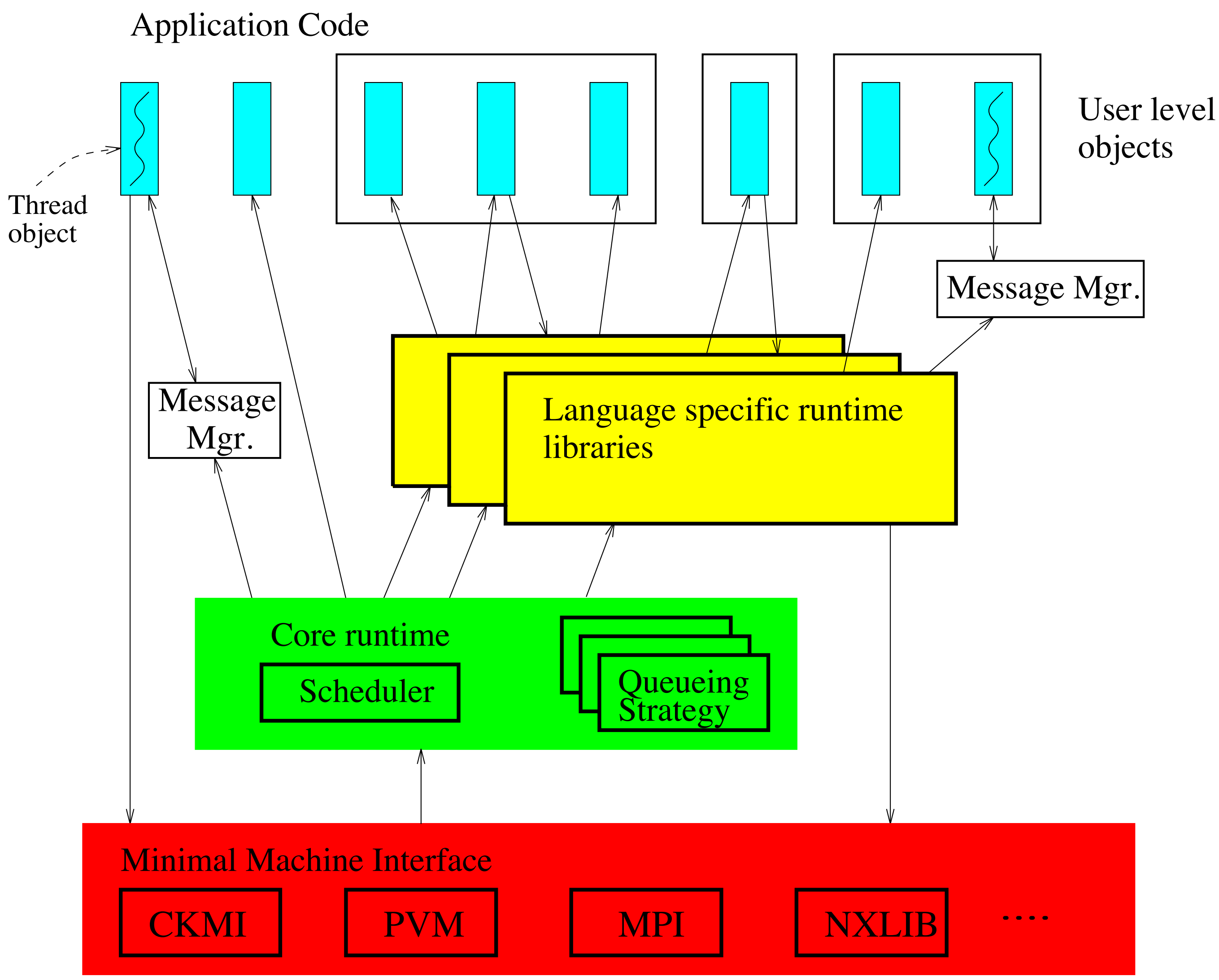
Control regimes who decides when control transfers between prog. components: *explicit and implicit*

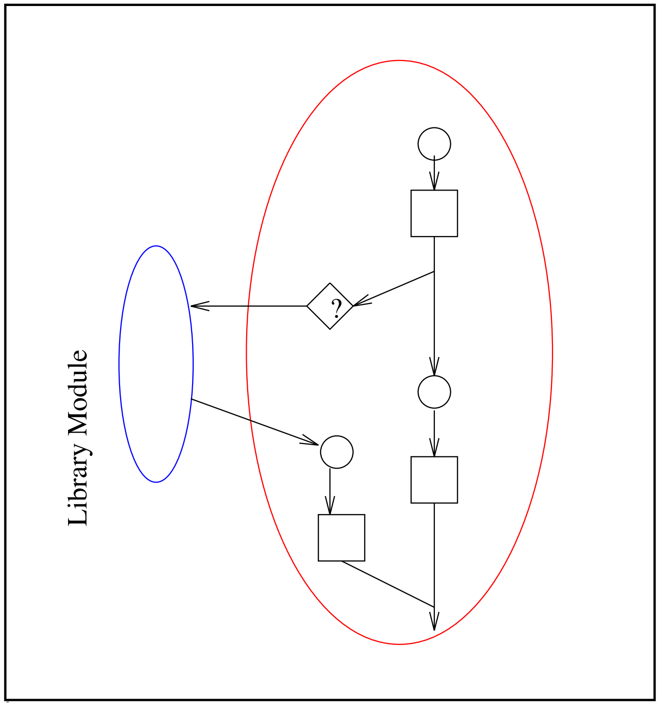
Entities in all (well..) languages can be classified as:

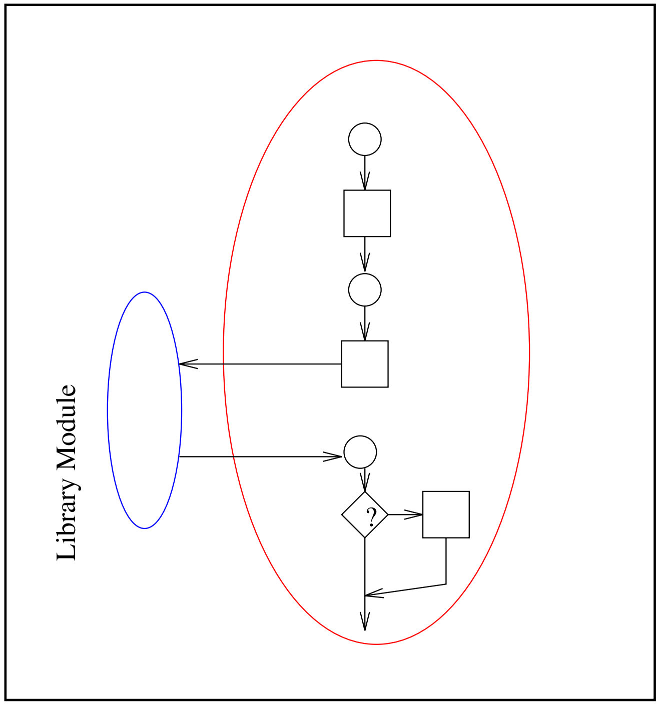
1. SPMD modules: no concurrency, Explicit control transfer
2. Threads: concurrency, implicit, limited stack
3. Message-driven Objects: concurrency, implicit

Converse: an interoperability framework

- Is implemented and available by ftp
- Currently allows modules from:
  - PVM, nxlib
  - PVM threads
  - Charm
  - Charm++
  - Charm + threads
  - DP
- Is a good framework for implementing your favorite language
- Feedbacks from language implementers sought


Application Code
User level objects
Thread object
Message Mgr.
Message Mgr.
Language specific runtime libraries
Core runtime
Scheduler
Queueing Strategy
Minimal Machine Interface
CKMI
PVM
MPI
NXLIB
....


Library Module
?


Library Module
?