Heuristic Approaches to Online Object Distribution

Pete Keleher    Bujor Silaghi
The Problem

• Target Environments
  – systems based on distributed objects
  – dynamic applications and environments (behavior + number)
  – no explicit information about applications
  – scalability ⇐ decentralized algs. + grain sizes

• Goal
  – efficient usage of:
    • the processors: load balance
    • the network: collocate related objects
  – graph-embedding problem, NP hard
Policies

• processor load estimation + overloaded and underloaded processors
  – simple: # objects, live, ready
  – profiled behavior: computation/communication, wasted cycles
    • object, class, processor
• load dissemination: append load info
• initial object placement: collocation, least loaded, underloaded
• object migration:
  – what: sender
  – where: target's host
  – when: invocations if: source/target load + object affinity ← invocation preference
• object eviction ← underloaded or overloaded hosts
  – not taken care of by migration
  – periodically
System

• scripting language with classes + sets of objects
• synthetic applications
• simulator: runtime system + processors + network
  – simplifying assumptions
• could use object traces: comp + comm phases
Results

- ultimate goal: performance
- load balancing alone: perfect balance
- with migration: balance sacrificed improvement = up to an order of magnitude
- load dissemination: similar to the oracle approach
- works in varying speed environments
- # used procs: proportional to degree of parallelism
Number of used processors as function of application parallelism
Conclusions

- comp/comm object profiling \quad processor state
- invocation profiling \quad object affinity
- migration: if informed, the greedier the better
- ready $\iff$ bad, live $\iff$ good
- enough communication \quad dissemination not a problem
- low overhead