Implicit Coordination in Clusters

David E. Culler

Andrea Arpaci-Dusseau

Computer Science Division

U.C. Berkeley

The Berkeley NOW Project

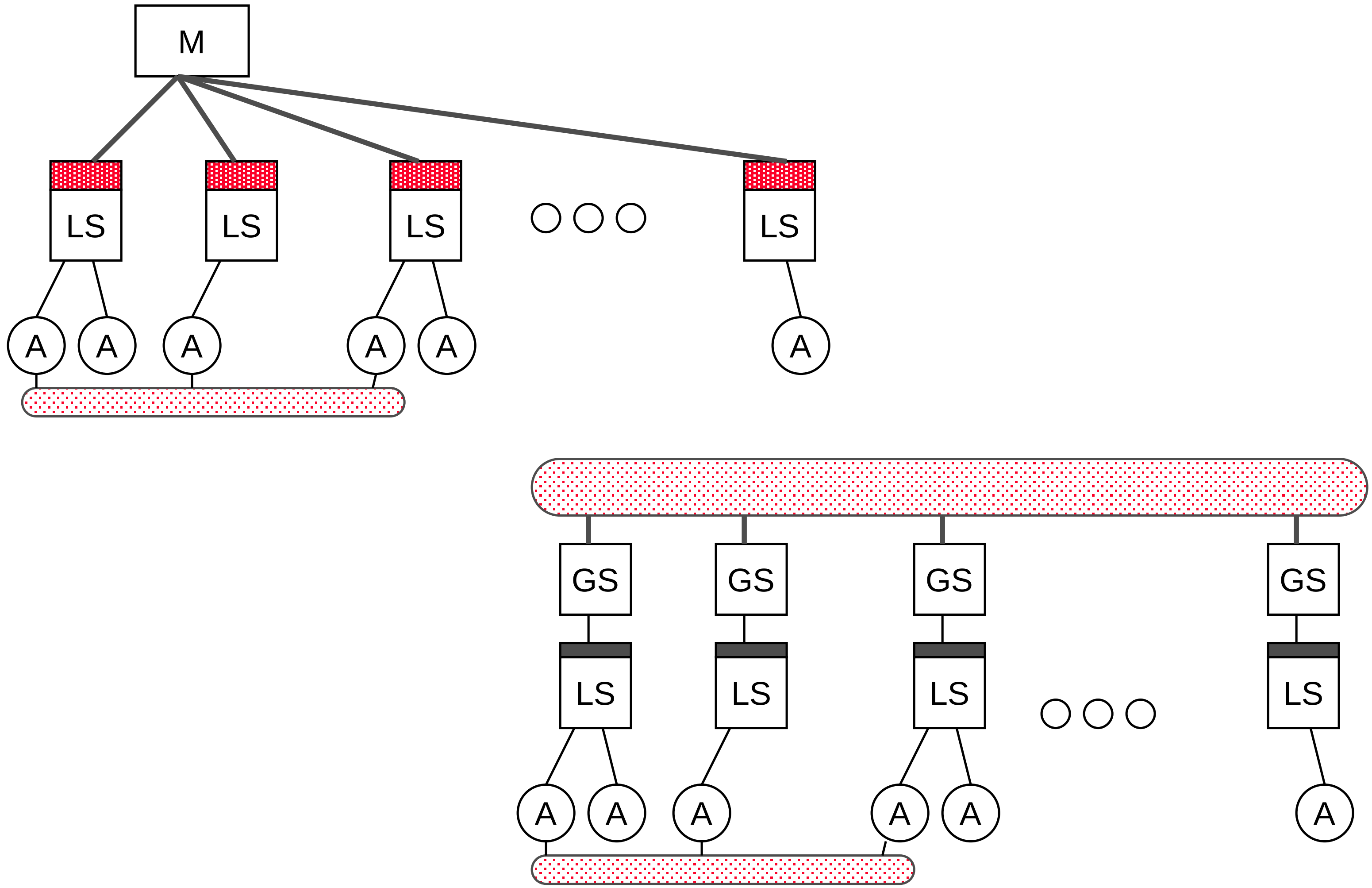
- **Large Scale: 100+ Sun Ultras, + PCs, +SMPs**
- **High-Performance**
  - **10-20 μs latency, 35 MB/s per node, GB/s aggregate**
  - **world leader in disk-to-disk sort, top 500 list, ...**
- **Operational**
  - **complete parallel programming environment**
  - **Glunix remote execution, load balancing, and partitioning**
- **Novel Technology**
  - **general purpose, fast communication with virtual networks**
  - **cooperative caching in XFS**
  - **clustered, interactive proportional share scheduling**
  - **implicit coscheduling**
- **Understanding of architectural trade-offs**

Clusters Means Coordination of Resources

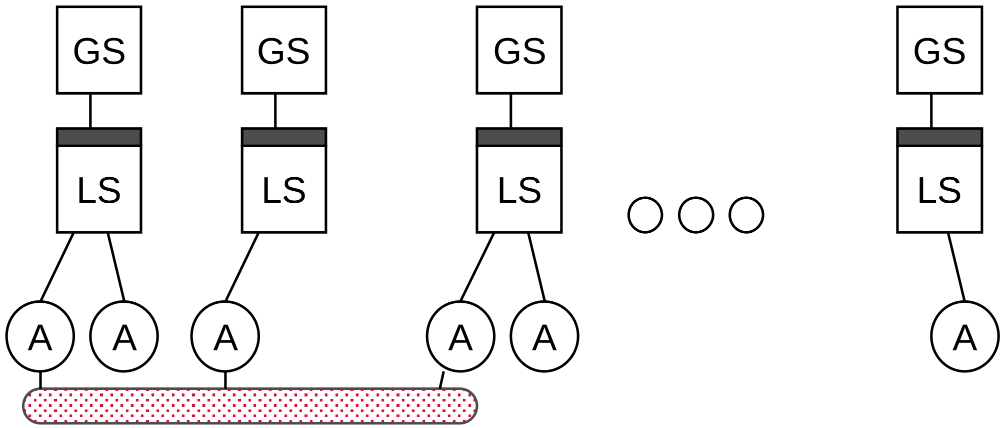
The Question

- **To what extent can resource usage be coordinated implicitly through events that occur naturally in applications, rather than through explicit subsystem mechanisms?**

Typical Cluster Subsystem Structures



How we'd like to build cluster subsystems



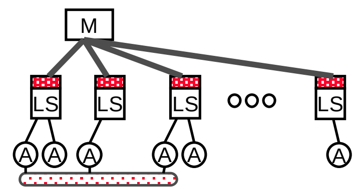
- **Obtain coordination without explicit subsystem interaction, only the events in the program**
  - very easy to build
  - potentially very robust
  - inherently "on-demand"
  - scalable
- **Local component can evolve**

Example: implicit coscheduling of parallel programs

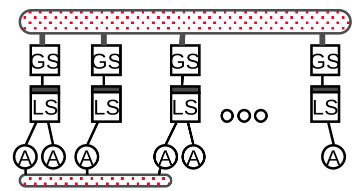
- **Parallel program runs on a collection of nodes**
  - **local scheduler doesn't understand that it needs to run in parallel**
  - **slow-downs relative to dedicated on-at-time execution huge!**

  => co-schedule (gang schedule) parallel job on the nodes

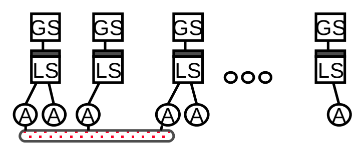
- **Three approaches examined in NOW**



  - **GLUNIX explicit master-slave (user level)**
    - » **matrix algorithm to pick PP**
    - » **uses stops & signals to try to force desired PP to run**



  - **explicit peer-peer scheduling assist**
    - » **co-scheduling daemons decide on PP and kick the solaris scheduler**



  - **implicit**
    - » **modify the PP run-time library to allow it to get itself co-scheduled with standard scheduler**

Problems with explicit coscheduling

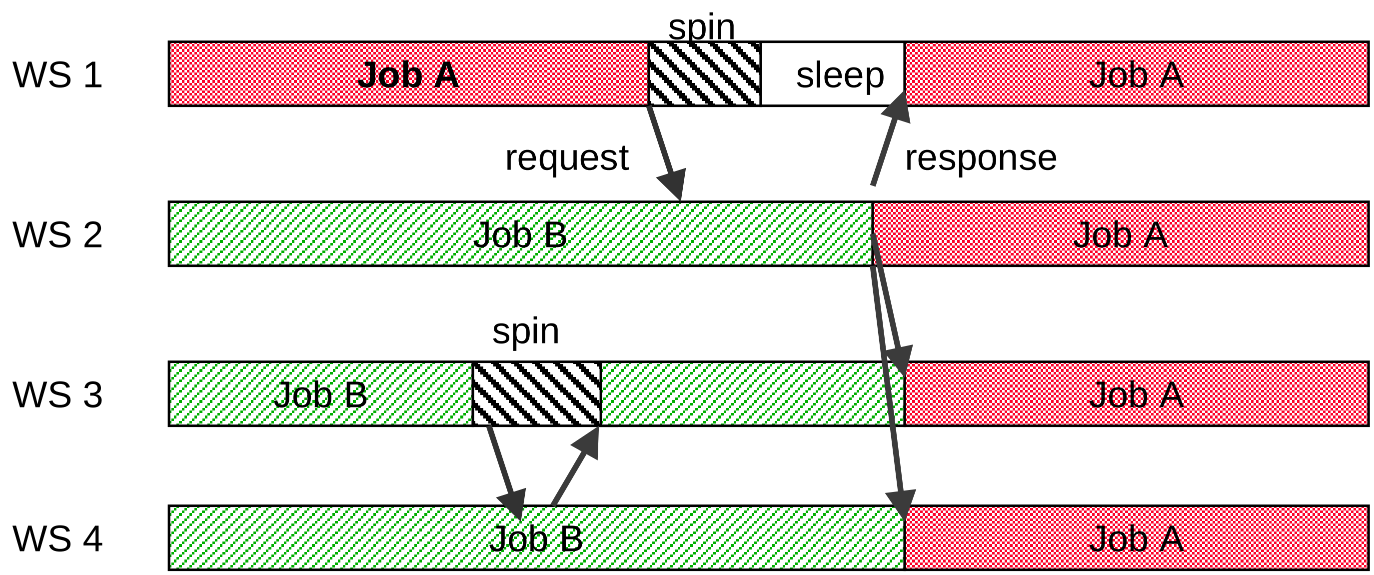
- Implementation complexity
- need to identify PP in advance
- interacts poorly with interactive use and load imbalance
- introduces new potential faults
- scalability

Why implicit coscheduling might work

- **Active message request-reply model**
  - like a read
- **Program issues requests and knows when reply arrives (local information)**
  - rapid response => partner probably scheduled
  - delayed response => partner probably not scheduled
- **Program can take action in response**
  - spin => stay scheduled
  - block => become unscheduled
  - wake-up => ???
    - » Priority boost for process when waiting event is satisfied means that it like to become scheduled while partner is still scheduled

Implicit Coscheduling

- **Application run-time uses two-phase adaptive-spin waiting for response**
  - **sleeps on AM event**
- **Solaris TS scheduler raises job priority on wake-up**
  - **may preempt other process**

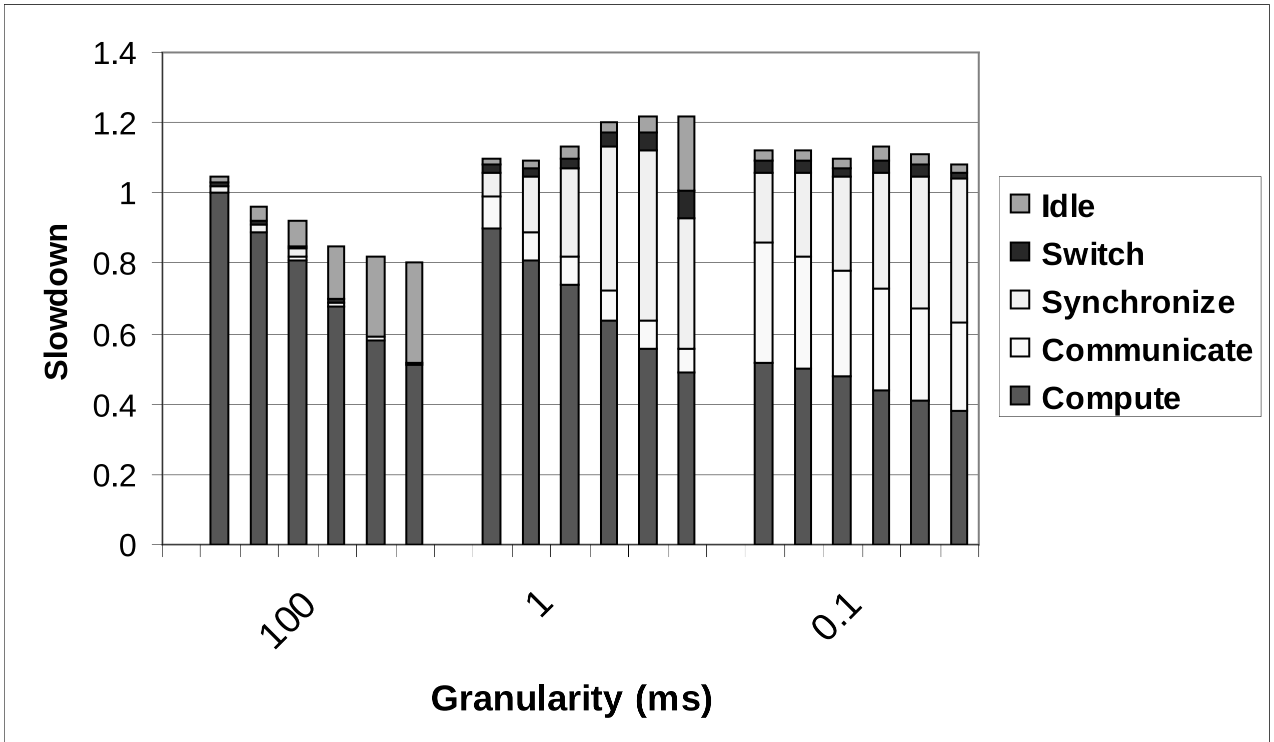


Obvious Questions

- **Does it work?**
- **How long do you spin?**
- **What are the requirements on the local scheduler?**

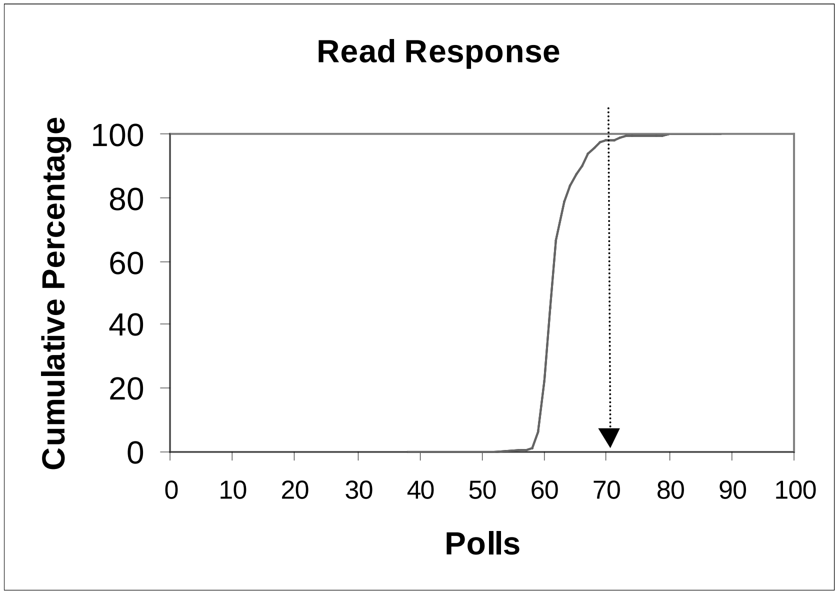
Simulation study

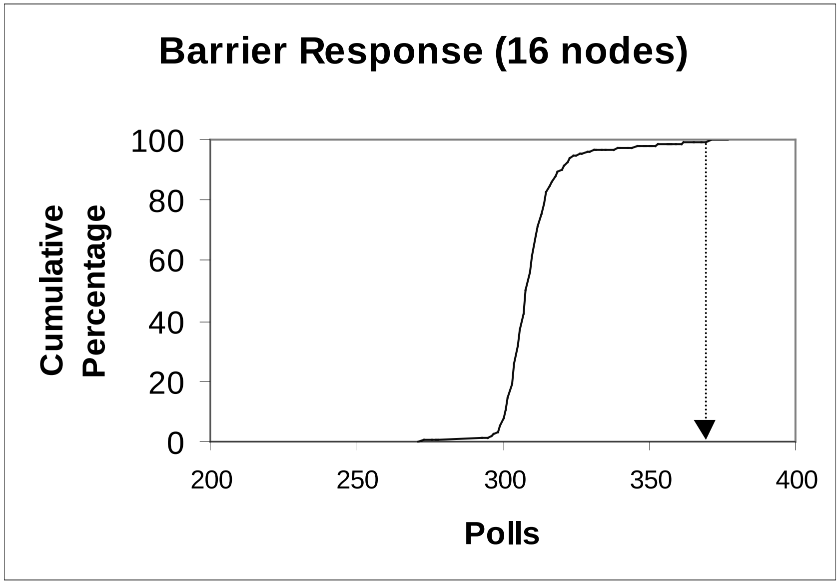
- **3 Parameterized synthetic bulk-synch. App'ns**
  - **communication pattern, granularity, load imbalance**
- **2-phase globally adaptive spin**
  - **round-trip time + load imbalance (up to 10 x ctx switch)**



Real world: how long do you spin?

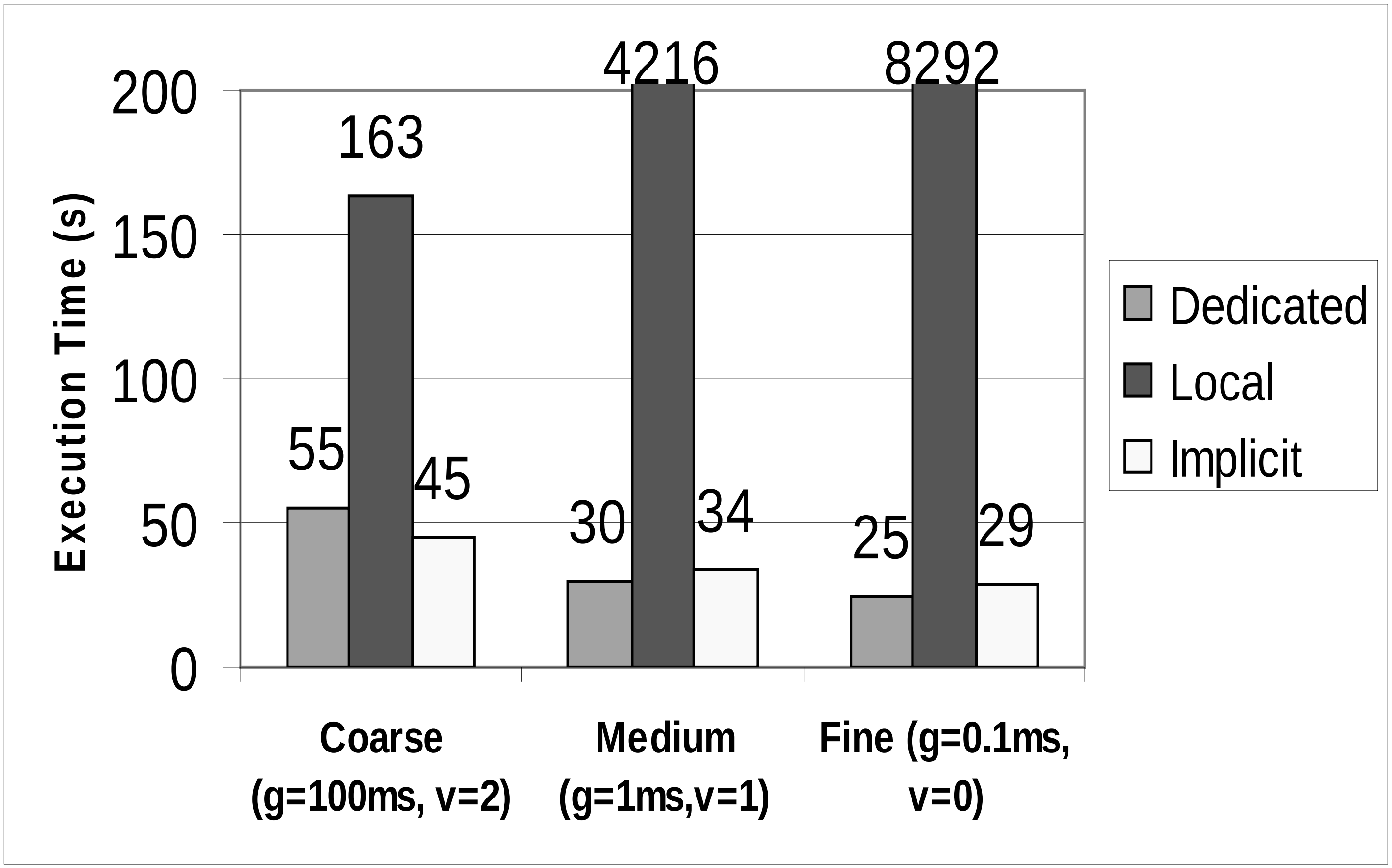
- **Use poll operation as basic unit**
- **Microbenchmark in dedicated environment**
  - get + synch: 140 polls
  - barrier: 380 polls





- **Barrier: spin for load imbalance (up to ~5 ms)**

How does it work?



Other implicit coordination successes

- Snooping based cache coherence
  - reading and writing data causes traffic to appear on the bus
  - cache controller observe and react to keep contents coordinated
  - no explicit cache-to-cache operations
- TCP window management
  - send data in bursts based on current expectations
  - observe loss and react
- AM NIC-NIC resynchronization
- Virtual network paging (???)
  - communicate with remote nodes
  - fault end-points onto NIC resources on miss
- ???

The Real Question

- How broadly can implicit coordination be applied in the design of cluster subsystems?
- **What are the fundamental requirements for it to work?**
  - **make local observations / react**
  - **local algorithm convergence toward common goal**
- **Where is it not applicable?**
  - **Competitive rather than cooperative situations**
    - » **independent jobs compete for resources but have no natural coupling that would permit observations**

Further reading

- **http://now.cs.berkeley.edu/**
- ***Extending Proportional-Share Scheduling to a Network of Workstations,*** **Andrea C. Arpaci-Dusseau, David E. Culler, International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), June, 1997.**
- ***Effective Distributed Scheduling of Parallel Workloads,*** **Andrea C. Dusseau, Remzi H. Arpaci, David E. Culler, SIGMETRICS '96.**
- ***The Interaction of Parallel and Sequential Workloads on a Network of Workstations,*** **SIGMETRICS '95 , 1995**