Application Level Scheduling of Gene Sequence Library Comparison for Metacomputing

Neil T. Spring and Rich Wolski

University of California, San Diego

ICS'98, July 15, 1998

Metacomputing

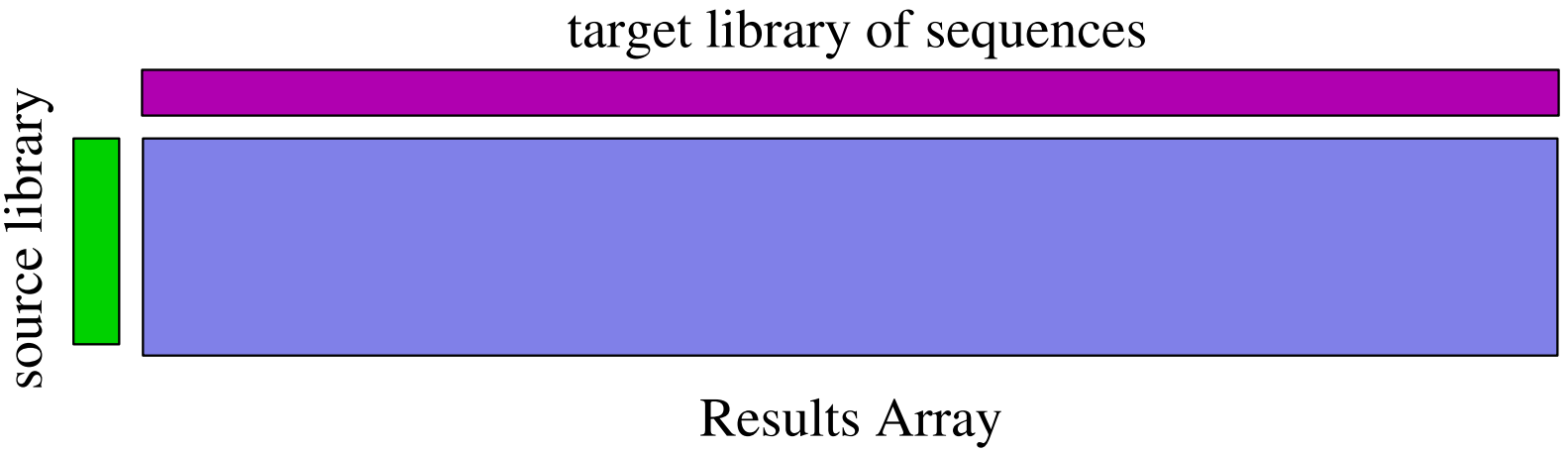
- **Almost all computational resources are networked**
  - Workstations, Supercomputers
  - Storage Devices
- **Increased network performance makes it possible to use distributed machines cooperatively**
- **How do you get performance? Scheduling.**

AppLeS

- **Metacomputer Application Scheduling**
  - Assignment of tasks and data to resources
  - Metacomputing environment
    - Distributed resources
    - Heterogeneity
    - Contention
- **Application Level Scheduling (AppLeS)**
  - Application specific control
  - Resources are seen in terms of application-specific performance

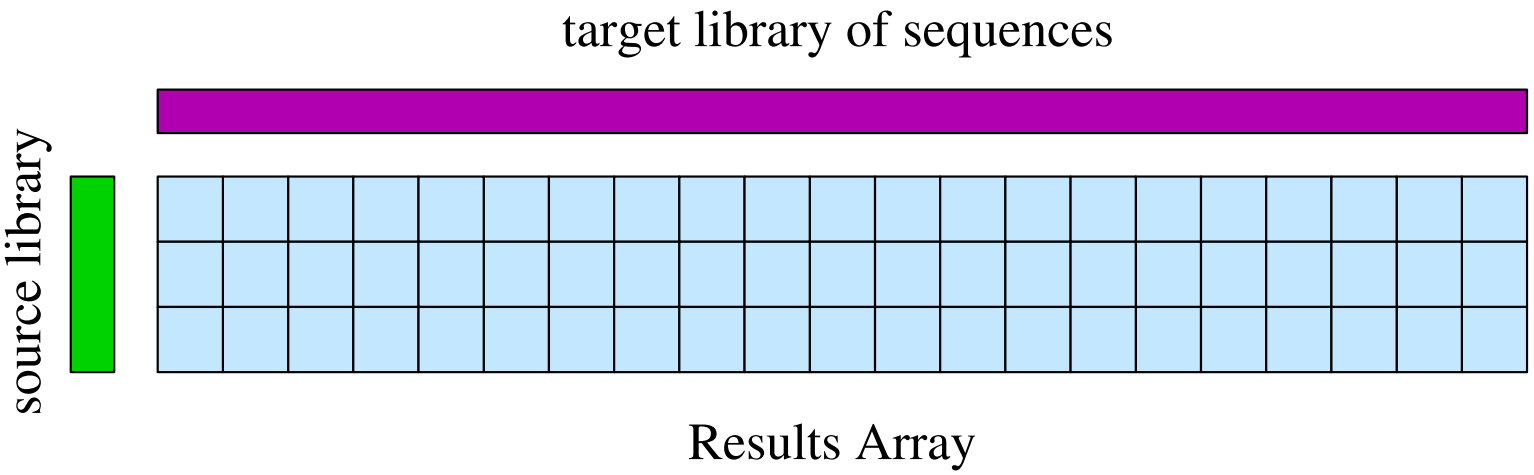
Gene Sequence Library Comparison

- **Biological sequences (proteins, DNA) are compared using a fast, sequential heuristic**
- **Comparison based on biological criteria**
  - Likelihood of various mutations
- **Matches indicate related structure, function, origin**



Parallel Library Comparison

- **More memory, more processor cycles**
- **Sequence comparisons are independent**
- **Divide and Conquer approach**
- **Chunks of sequence libraries are distributed to different hosts and compared in parallel**



Complib Implementation

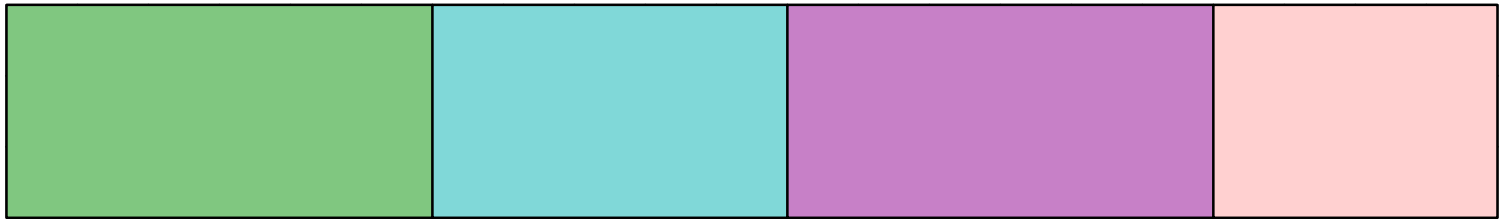
- **Sequences compared using FASTA**
  - Sequence comparison heuristic
  - Data dependent execution
- **Master/Slave structure:**
  - Master process distributes sequence library chunks
  - Slave processes compute the solution and return results to the master

Complib Scheduling Methods

- **Workstealing**
  - Idle “slaves” request new work from “master”
  - Size of work allocation determines load-imbalance
    - Tolerant of data dependent execution time
    - Tolerant of variance in processor performance
- **Static Placement**
  - Compute work placement before execution begins
  - Low overhead
  - Based on predicted execution time

Run-time Static Placement

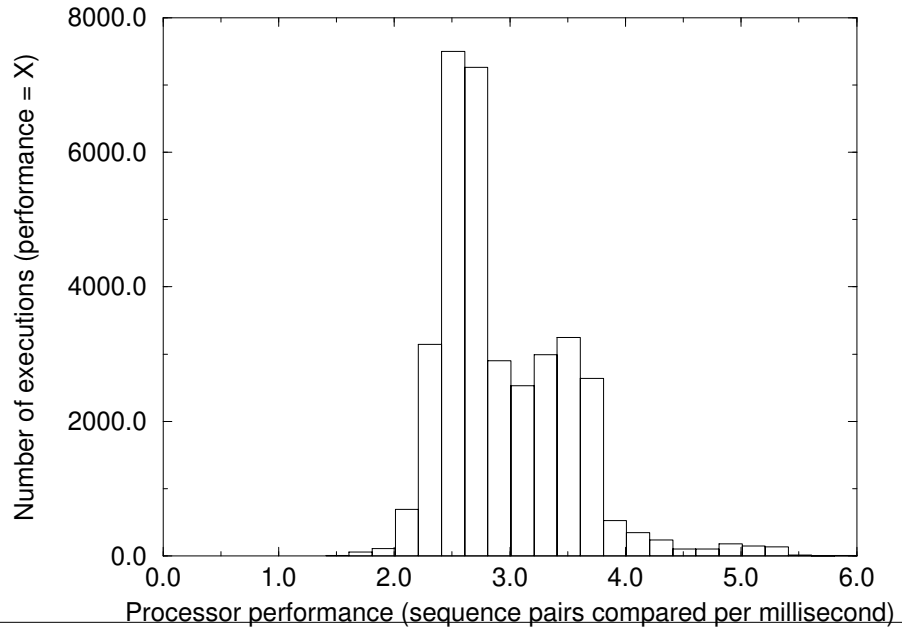
- **Time balancing: assign work so that all processors finish at the same time**
- **Execution time prediction for each slave**
  - Predict how long each task will take on each processor
  - Predict the "load" on each processor
- **Distribute the work in proportion to the relative capability of each processor**



AppLeS Static Strip Decomposition

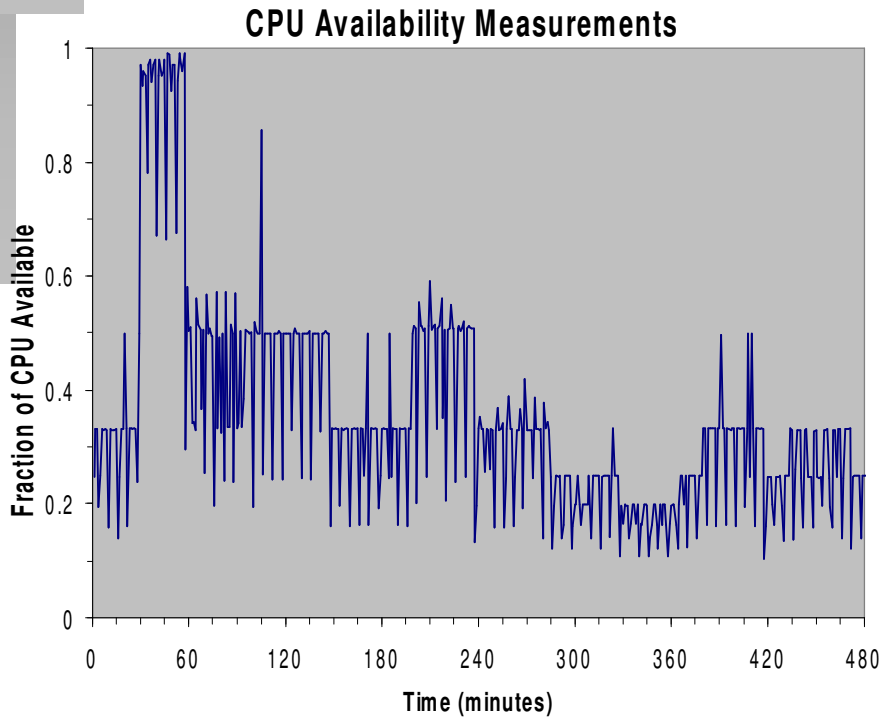
Data Dependent Execution

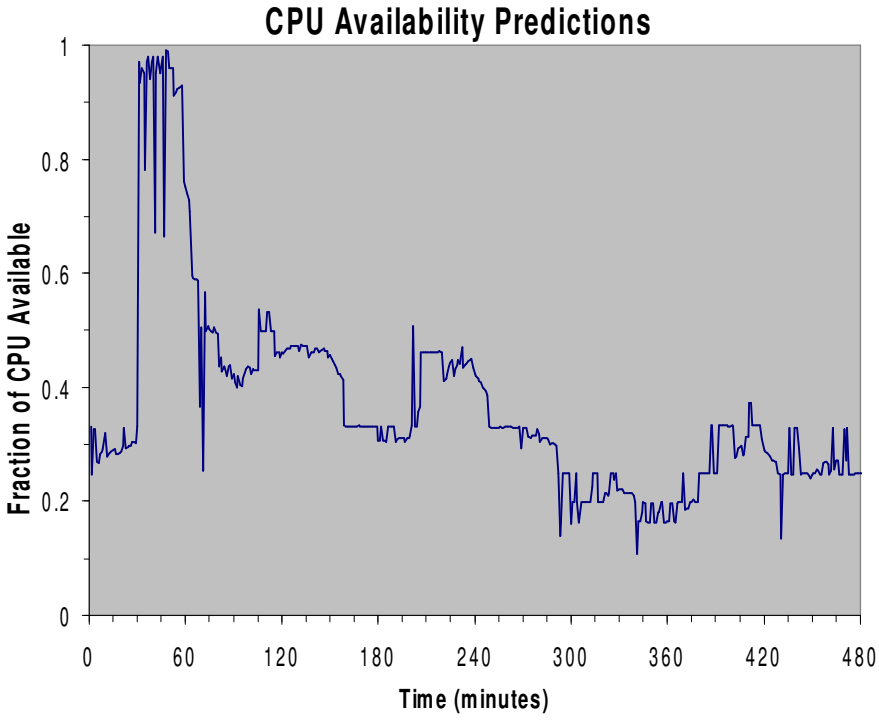
- **How long will it take to compare sequences even if there is no load?**
- **Data dependence ➡ Distribution of values**
- **Roughly symmetric ➡ Mean as representative**



Estimating load

- **How much of the processor will we get?**
- **Use a time series of measurements**
- **To generate near term predictions**





Problem with Run-time Static

- **Point value estimates are not always accurate**
- **Variation in execution time from:**
  - Data dependent execution
  - Changes in contention
- **Causes load imbalance**
- **Potential to offset reduction of overhead over workstealing**

Hybrid Scheduling Method

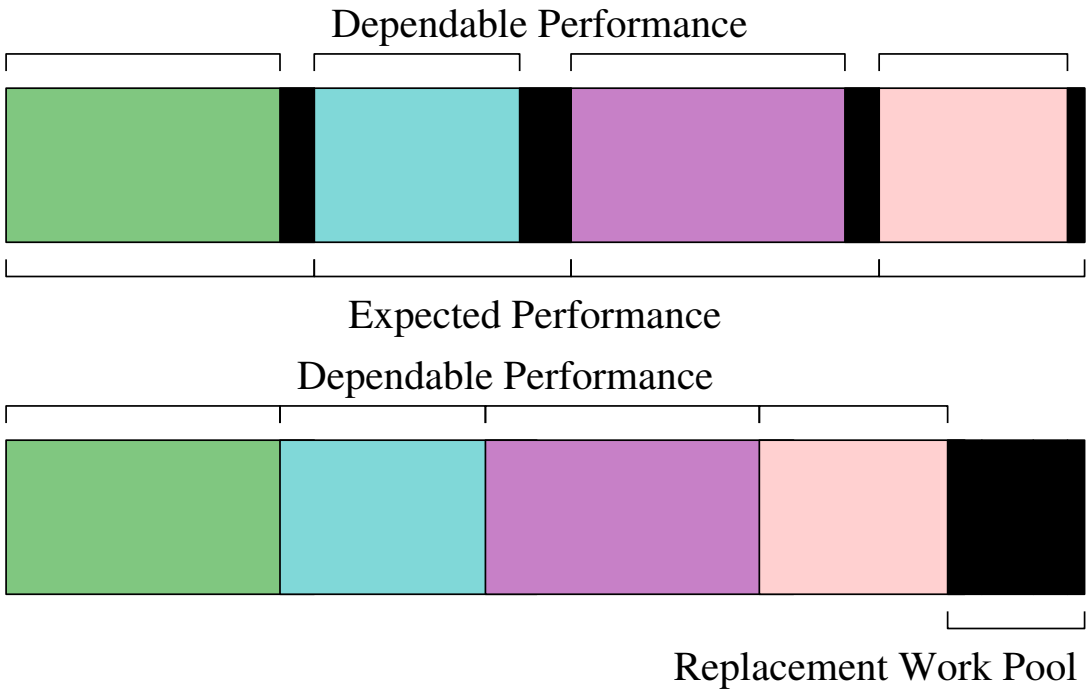
- **Reduce execution time using both:**
- **Placement (Run-time static)**
  - Master allocates work
  - Low communication & startup overhead
  - Place too much work ➡ load imbalance
- **Replacement (Workstealing)**
  - Slaves request additional work
  - Reduced load imbalance
  - Replace too much work ➡ excessive overhead
- **How much should be placed?**

Hybrid Scheduling Boundary

- **Expected Performance**
  - Number of sequences we expect to be able to compare before the application finishes
  - NWS Prediction & Application benchmark
- **Dependable Performance**
  - Artificial lower bound on performance
  - NWS Prediction & Application benchmark & Variance

Hybrid Scheduling

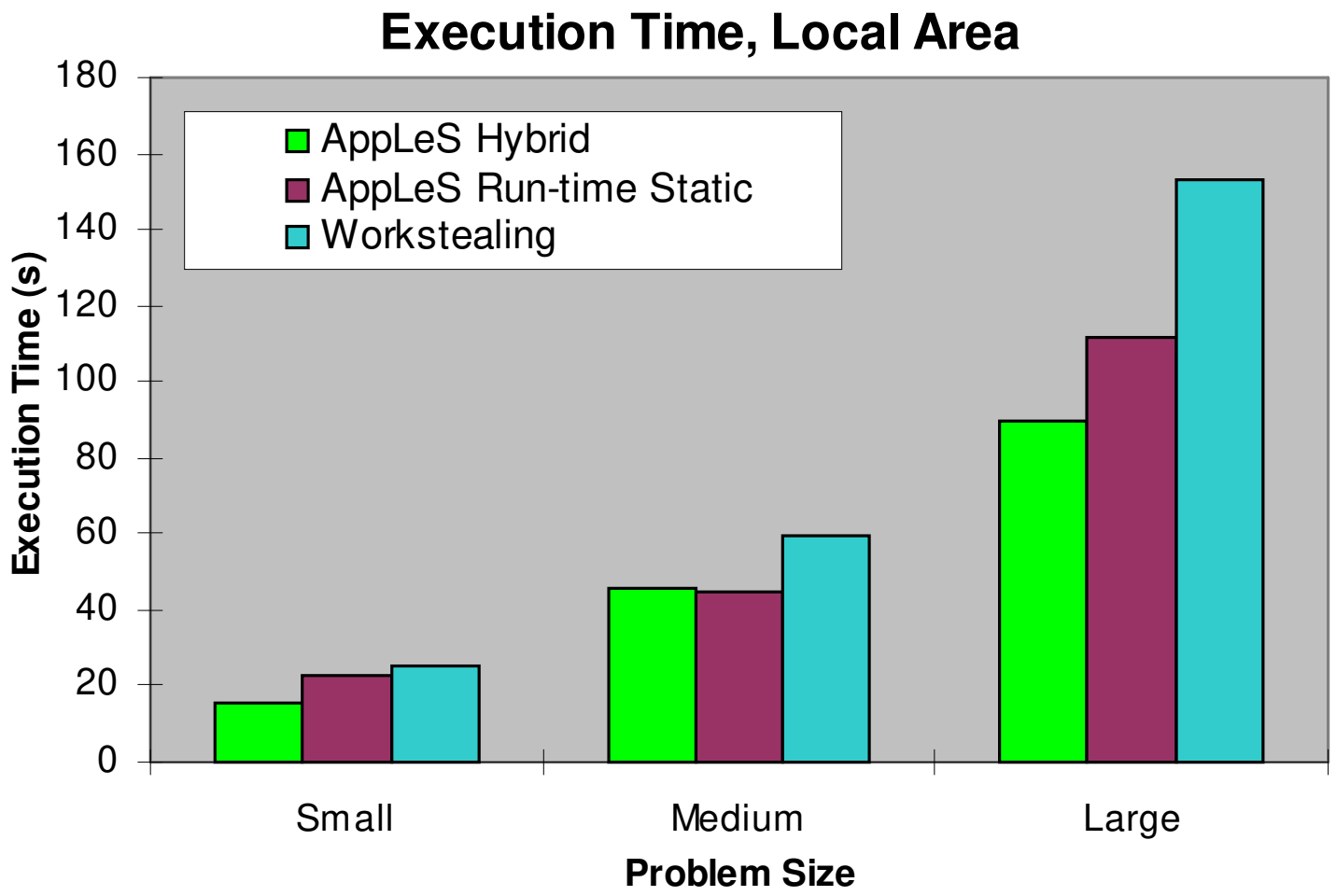
- **Work partitioned based on *dependable* and *expected* performance**
- **Colors represent placed processor work, black represents work deferred for replacement**



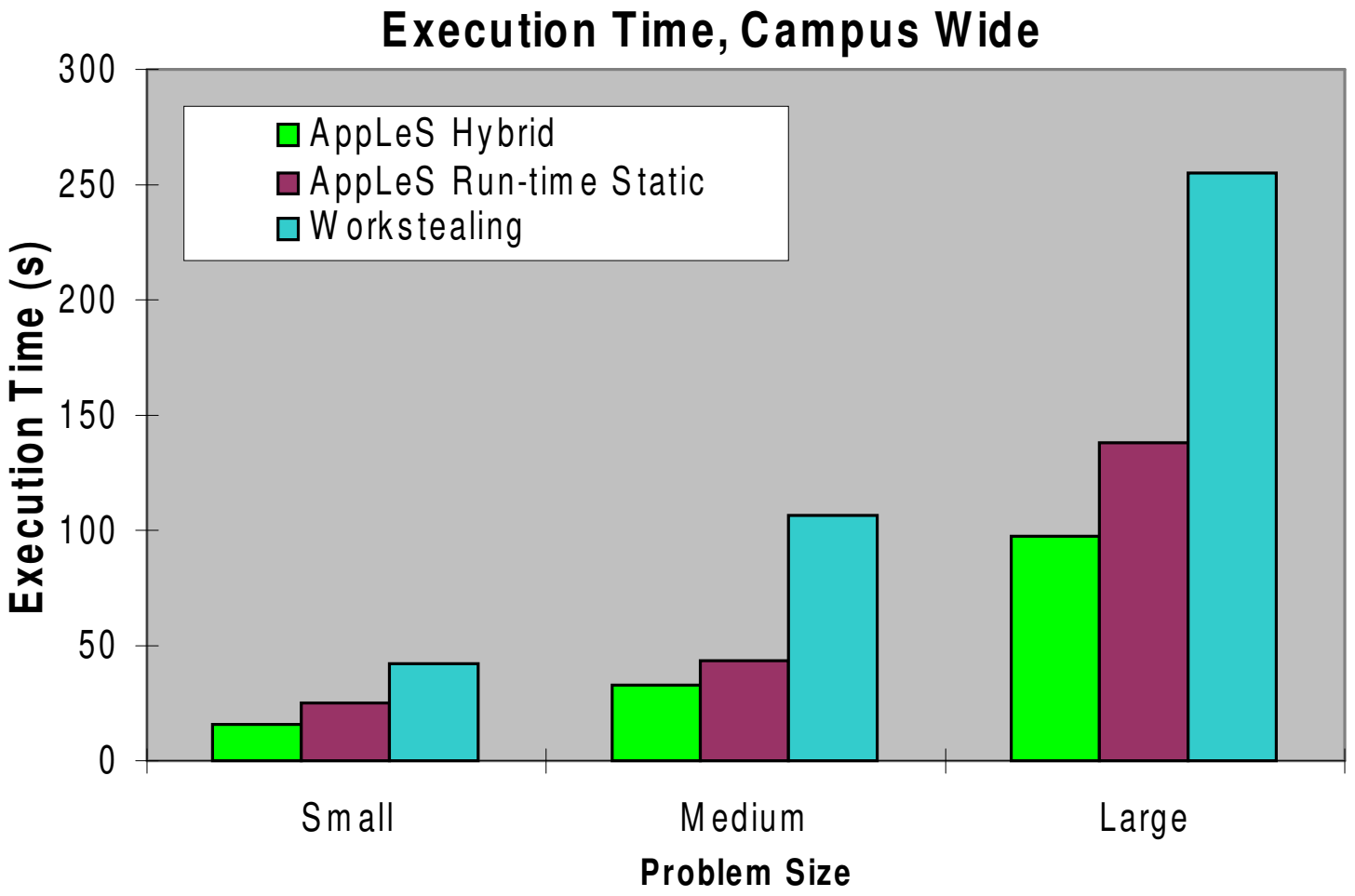
Comparing Scheduling Methods

- **Three scheduling methods**
  - Workstealing: Replacement only
  - Run-time Static: Placement only
  - Hybrid: Placement & Replacement
- **Three Platforms**
  - Local
    - SDSC HPC Cluster & 3 Workstations (25 processors)
  - Campus
    - Adds PCL Suns (4) by campus backbone
  - National
    - Adds CS6400 (12 processor) via cross-country ATM
    - And additional SDSC HPC machines (8 processors)

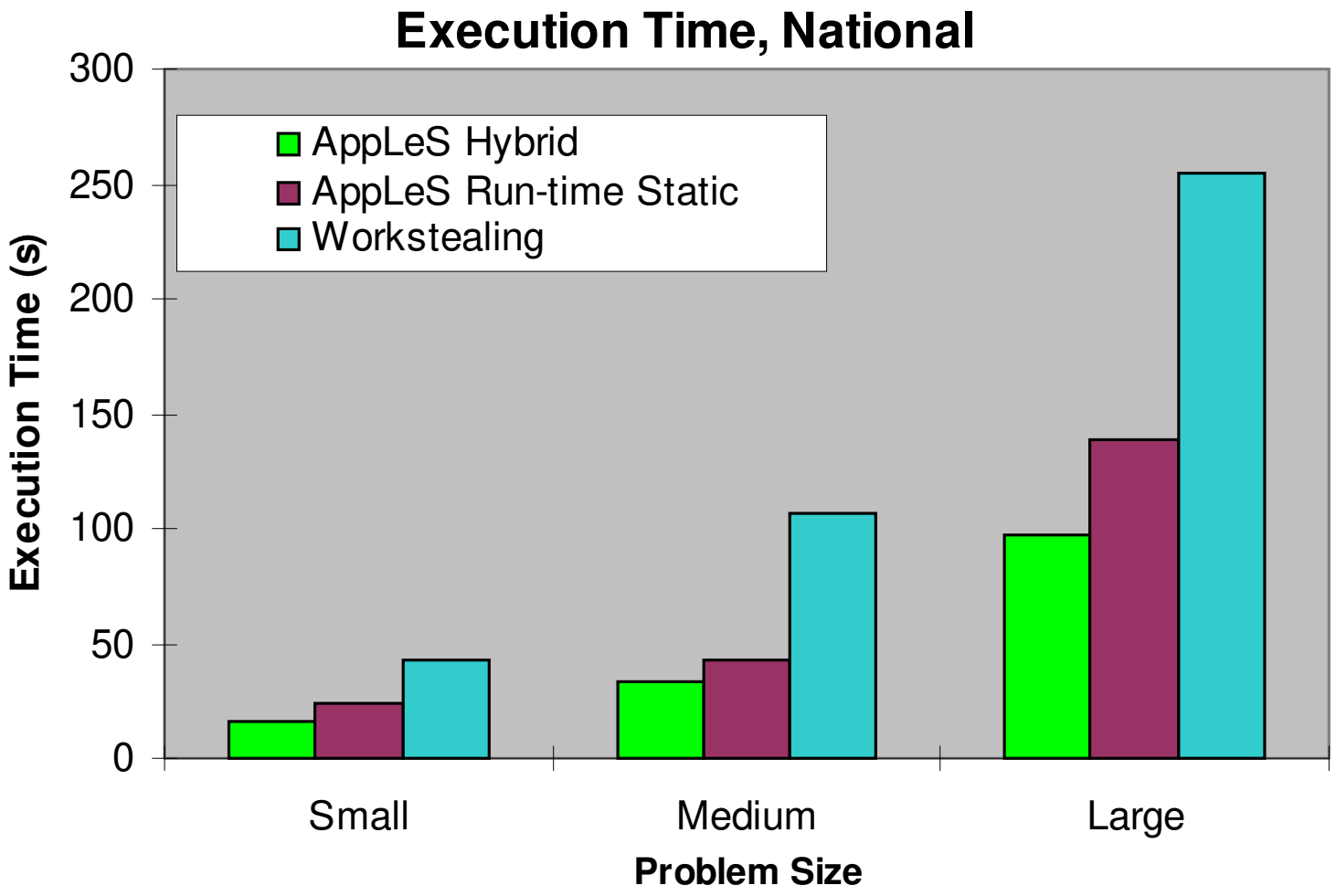
Results: Local Area



Results: Campus Wide



Results: National



Conclusions

- **Possible to make predictions of data dependent execution time in a shared environment.**
- **Scheduling methods must tolerate inaccuracies necessarily part of predictive process**
- **Using predictions & accuracy measures we can combine scheduling policies for more efficient overall schedules**

Contact Information

- **Neil Spring**

  nspring@cs.ucsd.edu
- **Rich Wolski**

  rich@cs.ucsd.edu
- **AppLeS**

  http://www.cs.ucsd.edu/groups/hpcl/apples
- **Network Weather Service**

  http://nws.npaci.edu