Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids

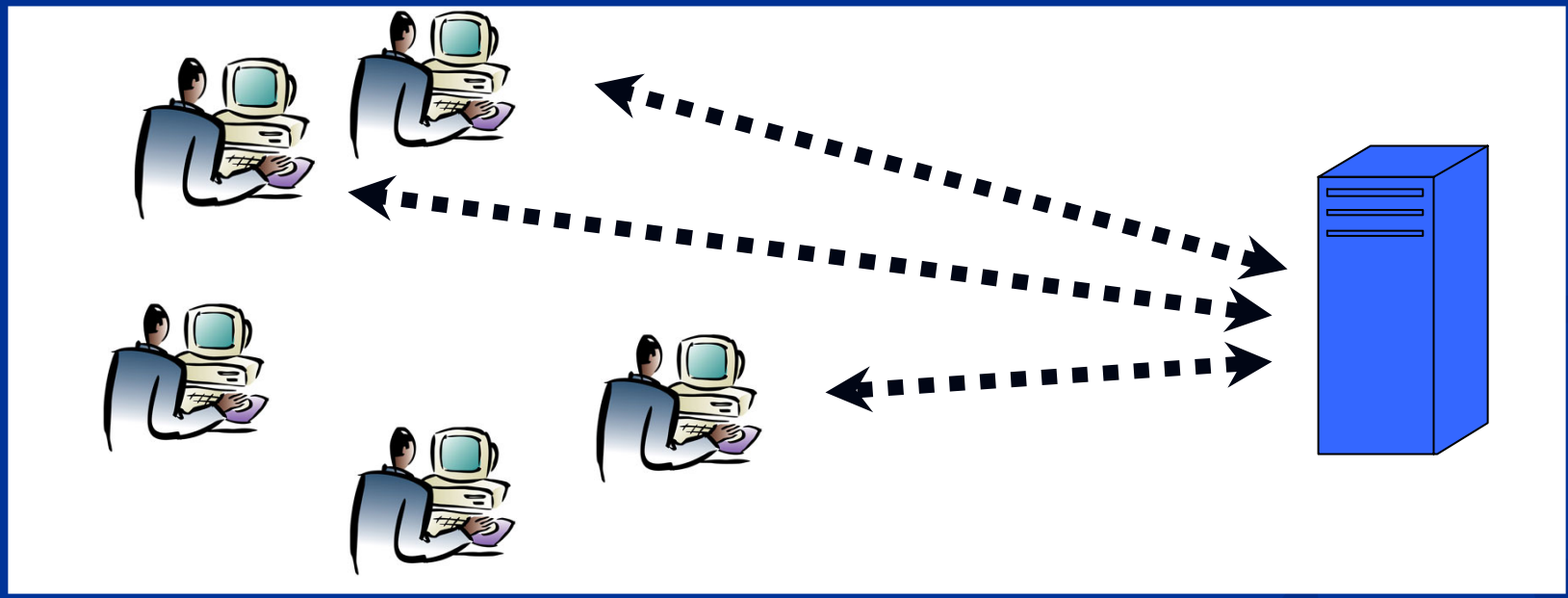
Derrick Kondo†, Andrew A. Chien†, Henri Casanova†‡

Computer Science and Engineering Department † San Diego Supercomputer Center ‡

University of California at San Diego

Desktop Grid Background

- Set of many network-connected computing/storage resources



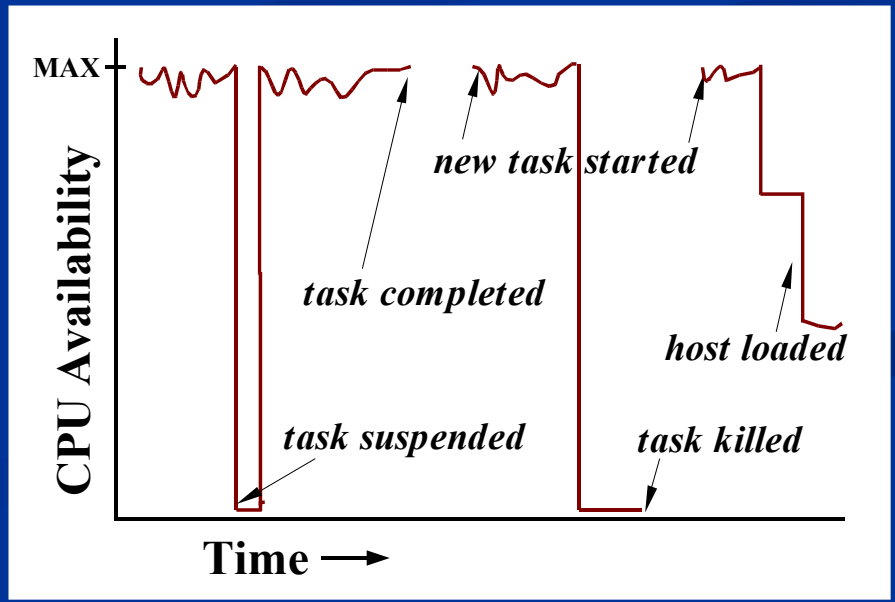
- Resources: heterogeneous & shared

Motivation

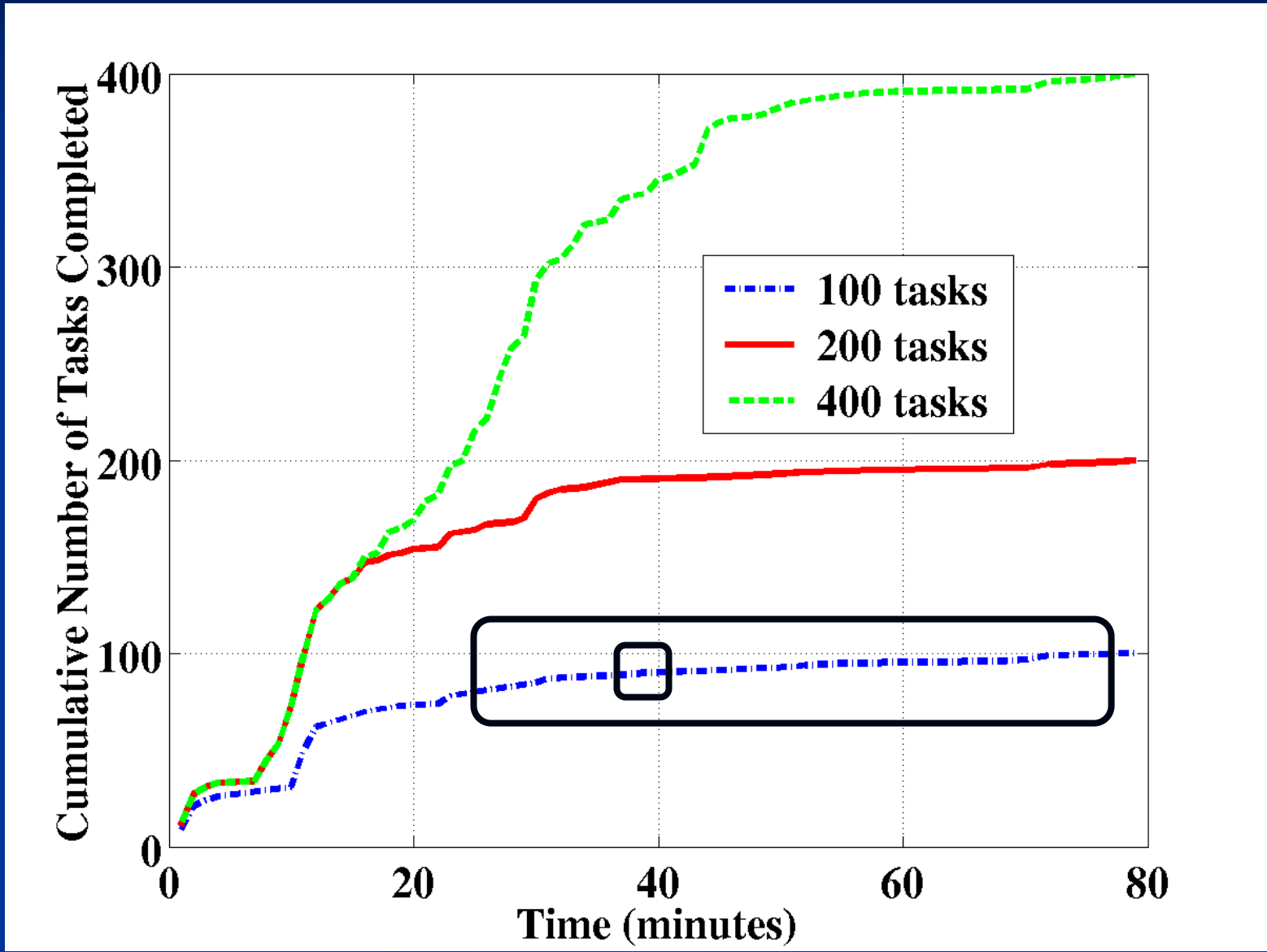
- High computational power at low cost
  - Reuse existing infrastructure of resources
- Successful deployment of high-throughput, compute-intensive applications
  - E.g. SETI@home, folding@home, fightaids@home
- Applications (especially in industry) require O(hour) turnaround time

Challenge

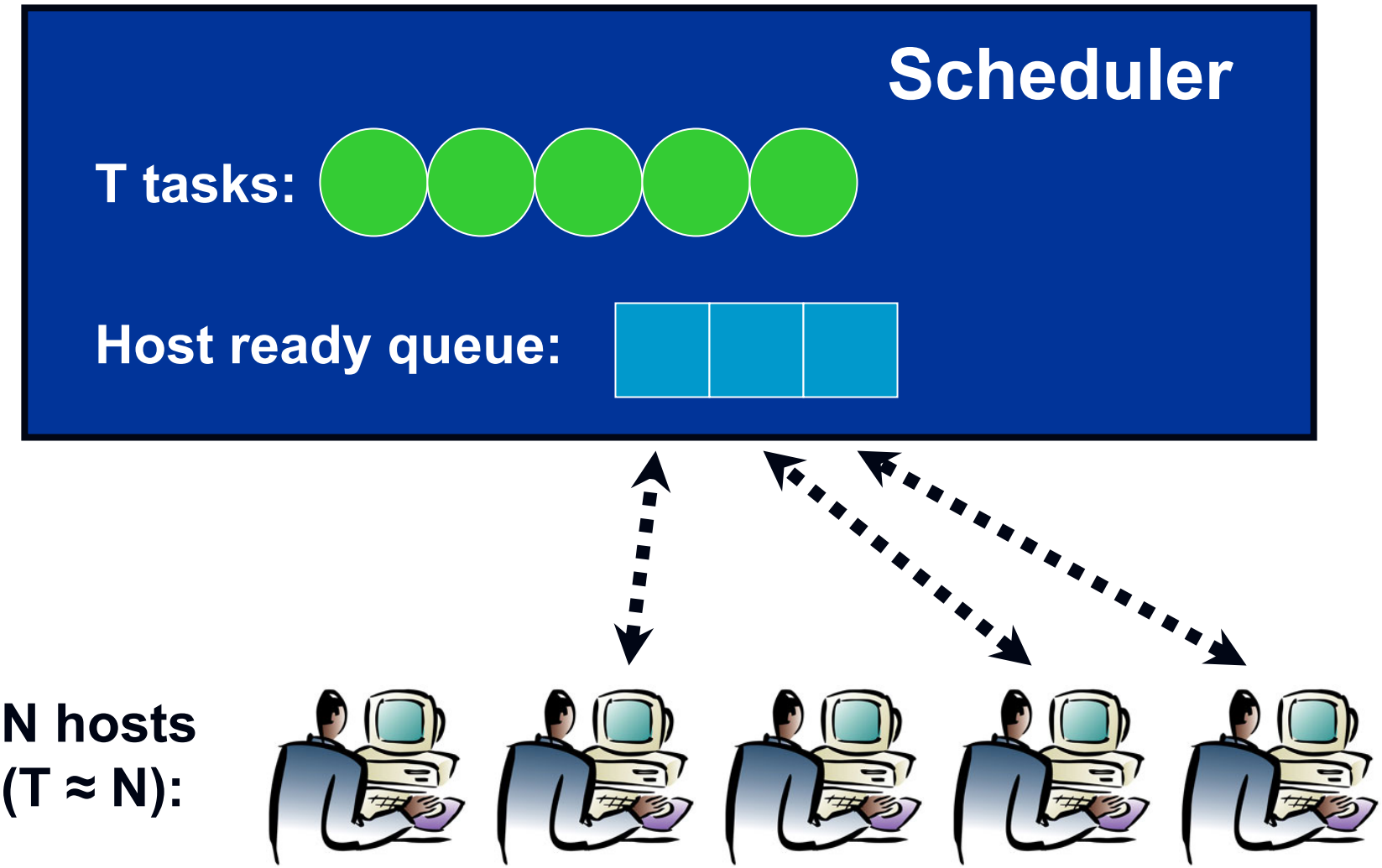
- Support rapid turnaround for applications on volatile desktop grids
  - Resources are heterogeneous: e.g. wide range of clock rates
  - Resources shared: unreserved, volatile
    - Variable CPU load
    - Variable host availability



Cumulative Task Completion For FCFS



Scheduling Problem



Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

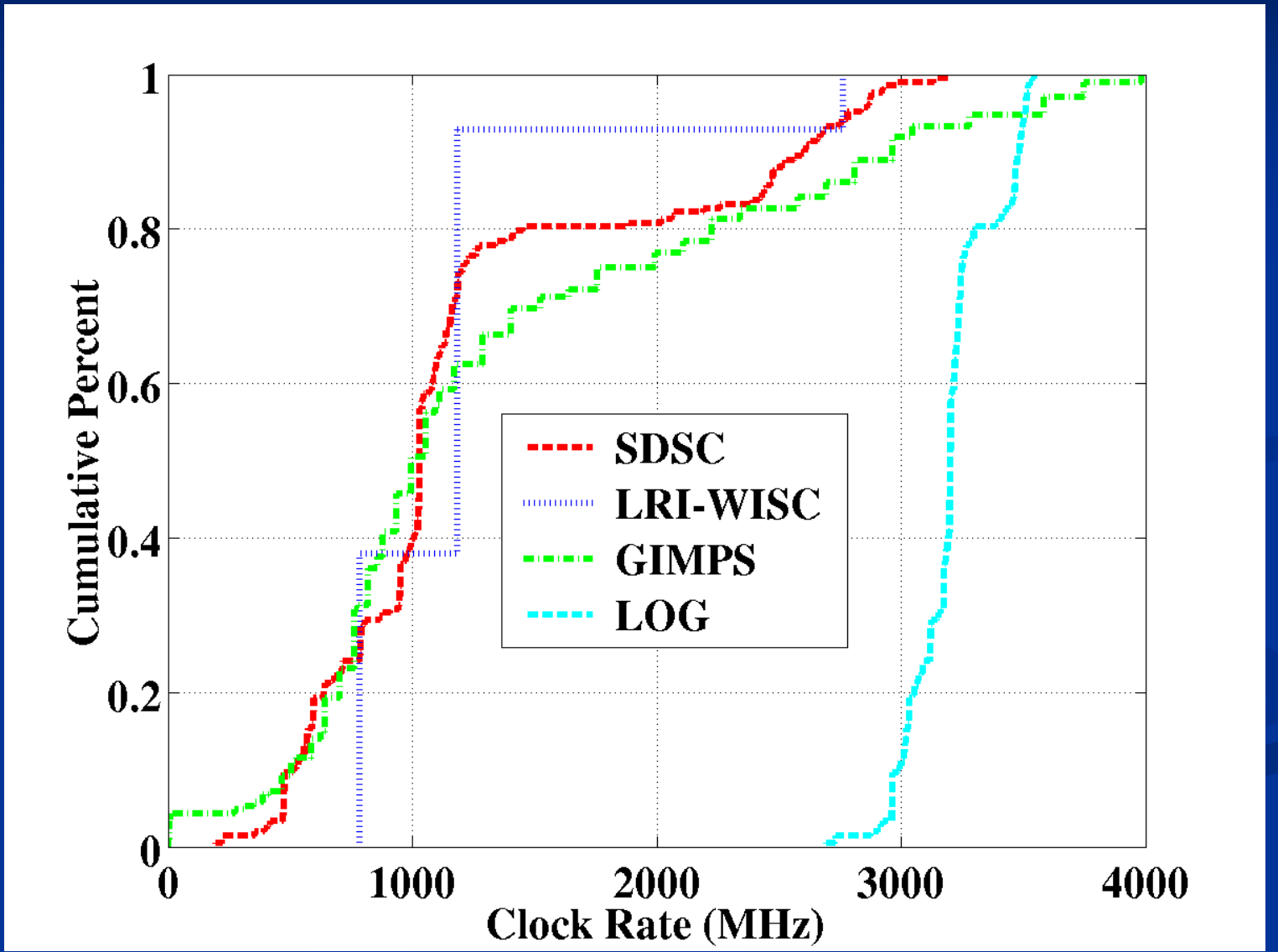
Simulation Method

- Used trace-driven simulation
  - Traces from a real desktop grid
  - Simulated 3 other representative desktop grid platforms
    - Homogeneous grid, Multi-cluster grid, and Internet grid
- Evaluated heuristics for scheduling short-lived applications, varying task size and number

Availability Traces

- ~230 desktops at the San Diego Supercomputer Center (SDSC) running the Entropia desktop grid software
- Submitted infinite workload of tasks to Entropia
  - Task continuously computed a mix of floating point/integer operations and wrote number completed every 10 secs to file
  - Tasks did not interfere with desktop user
- Obtained traces for a cumulative period of 1-month
- **Observe host and CPU availability exactly as any real desktop grid application would**

Clock Rate Distributions



Simulated Applications

- Task sizes
  - 5, 15, 35 minutes on dedicated 1.5GHz host
  - Corresponding failure rates of 6.33%, 13.2%, 22%
- Task number
  - 100, 200, 400 tasks
- For each task number and size, chose random starting points during business hours
  - Conducted over 100 experiments per task number and size
- Performance metric: ratio of makespan relative to optimal
  - Optimal makespan computed using omniscient scheduler with full knowledge of the future through the traces

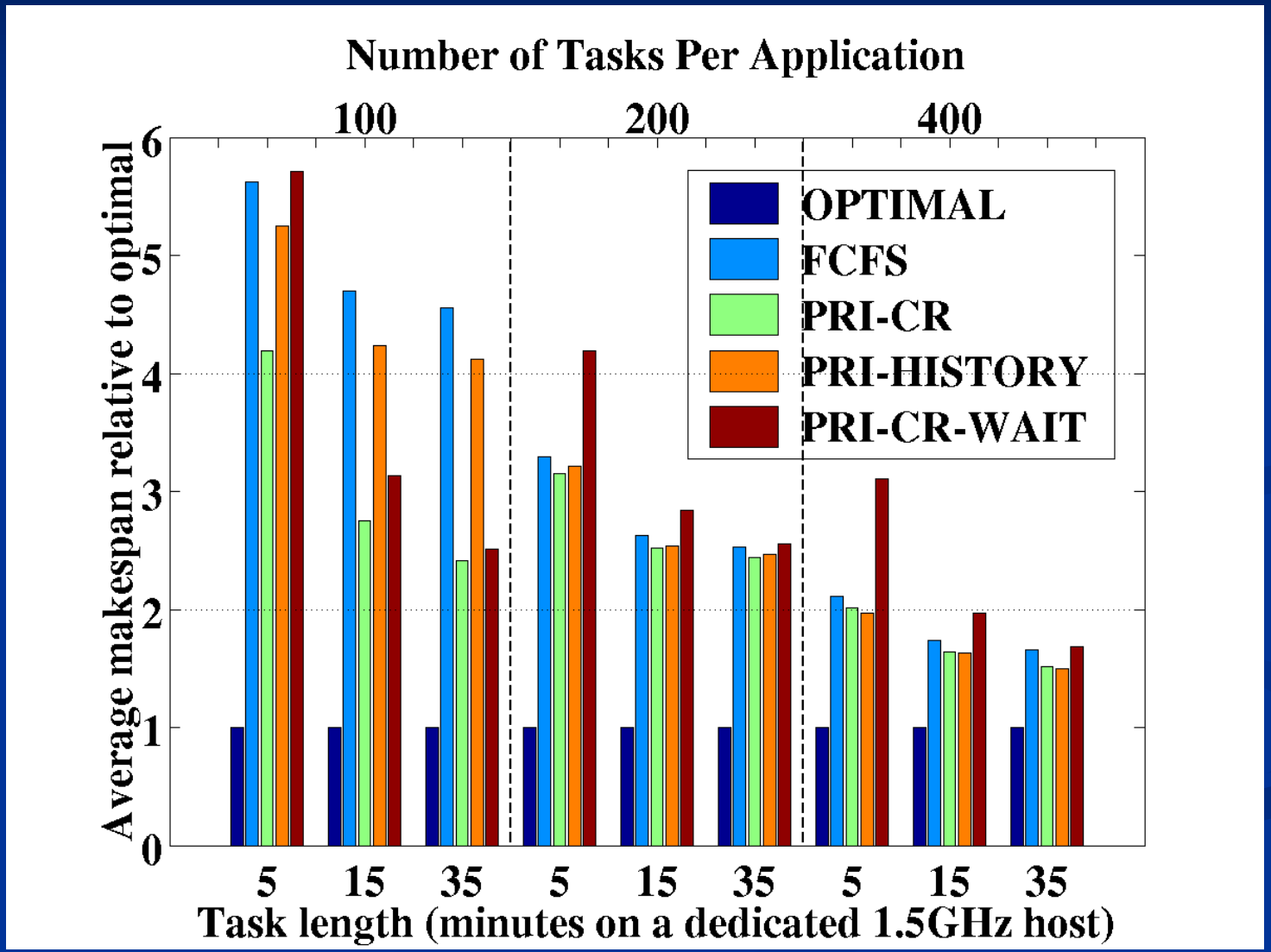
Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

Resource Prioritization

- Prioritize hosts in ready queue to use "better" hosts earlier
- PRI-CR
  - Use clock rates
- PRI-CR-WAIT
  - Use clock rates but wait for fixed period of time (10 min)
- PRI-HISTORY
  - Use expected operations per interval

Performance on SDSC Grid



Conclusions

- If T >= N
  - Little benefit of prioritization over FCFS
    - FCFS: best hosts are assigned the most tasks
    - Prioritization when there are far more tasks than hosts or far fewer tasks than hosts is not beneficial
  - Waiting for available hosts decreases performance
- If T< N
  - CR works as well as or better than PRI-HISTORY
  - Prioritization resulting in exclusion can improve performance
    - PRI-CR on average 1.65 times better than FCFS for applications with 100 tasks

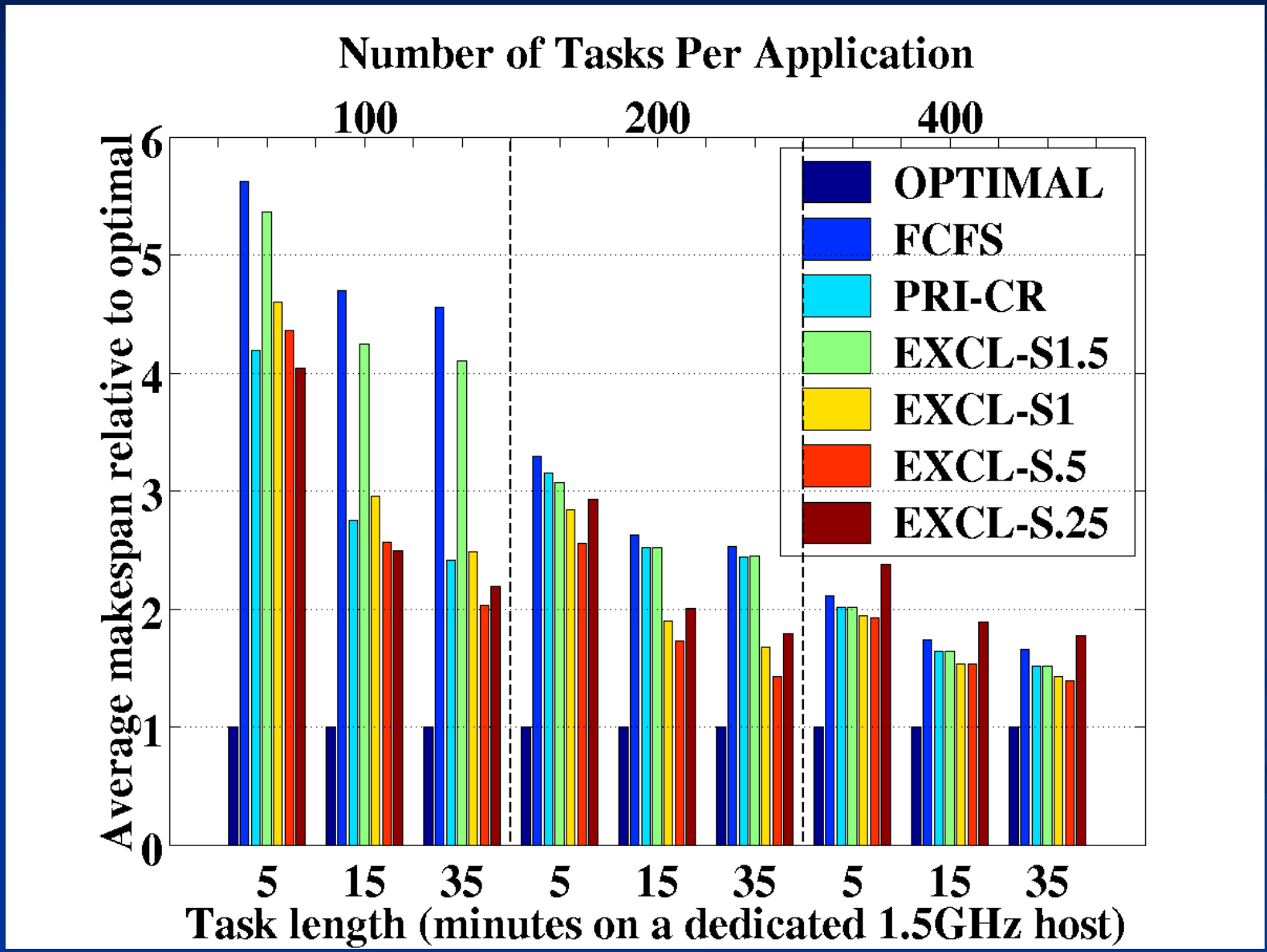
Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

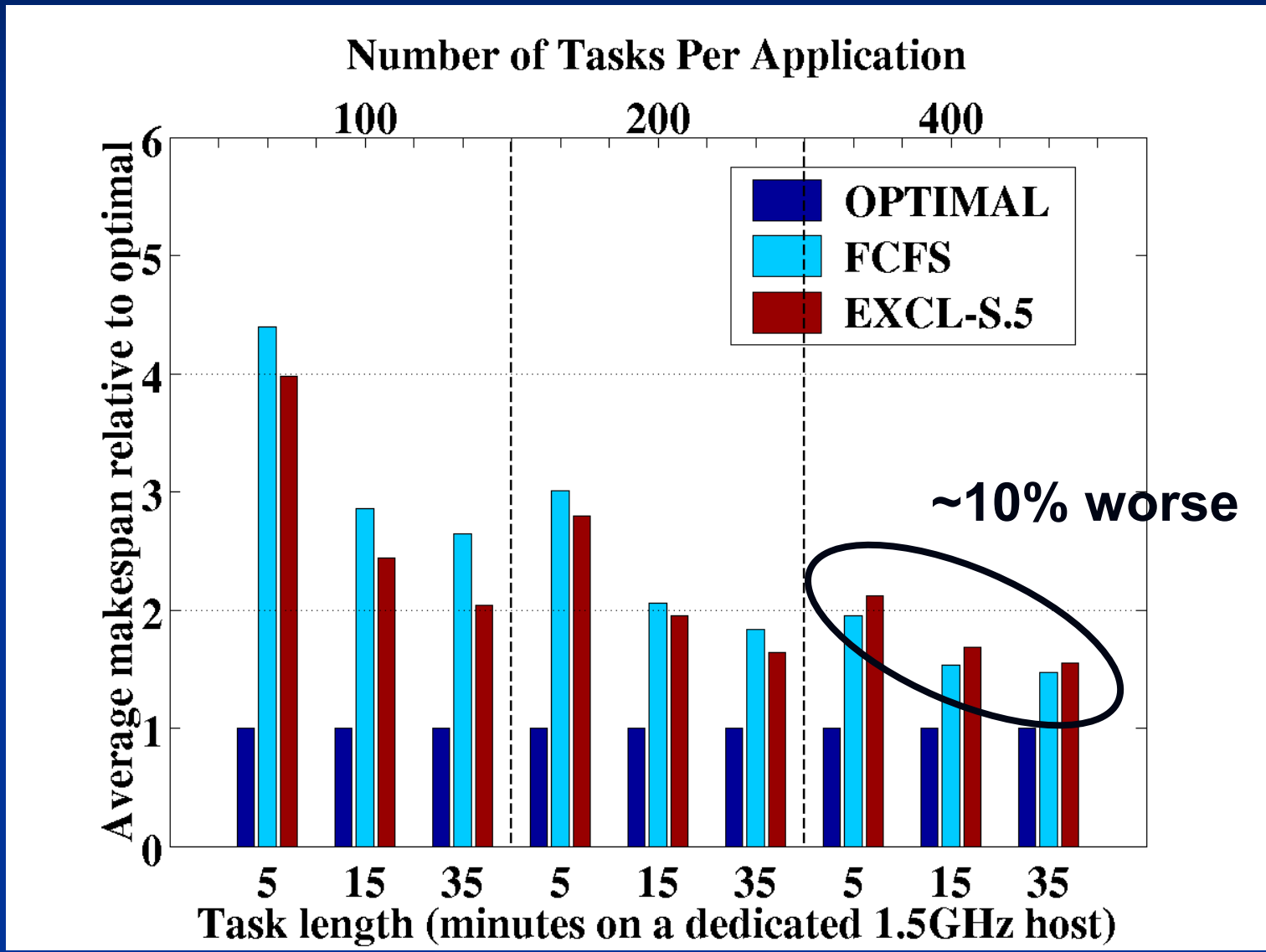
Resource Exclusion Using Clock Rates

- Exclude "worse" hosts to avoid slow task execution
- Use threshold that is some standard deviations below the mean clock rate
  - EXCL-S1.5
  - EXCL-S1
  - EXCL-S.5
  - EXCL-S.25

Performance Using Thresholds on SDSC Grid



Performance Using EXCL-S.5 on LRI-WISC Grid



Conclusions

- Resource exclusion can be significantly beneficial
  - EXCL-S.5 is 1.49 times better than FCFS on SDSC grid
- Exclusion using a fixed threshold can sometimes degrade performance, depending on the distribution of host speeds

Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

Resource Exclusion Using Predicted Makespans

- Predict application makespan and exclude those resources unable to complete task by projected completion time
- To predict makespan
  - Compute average operations completed per second (***r***)
  - Optimal execution time = ceil(***T*** / ***N***) * (***s*** / ***r***)

    where ***N*** is total number of hosts,

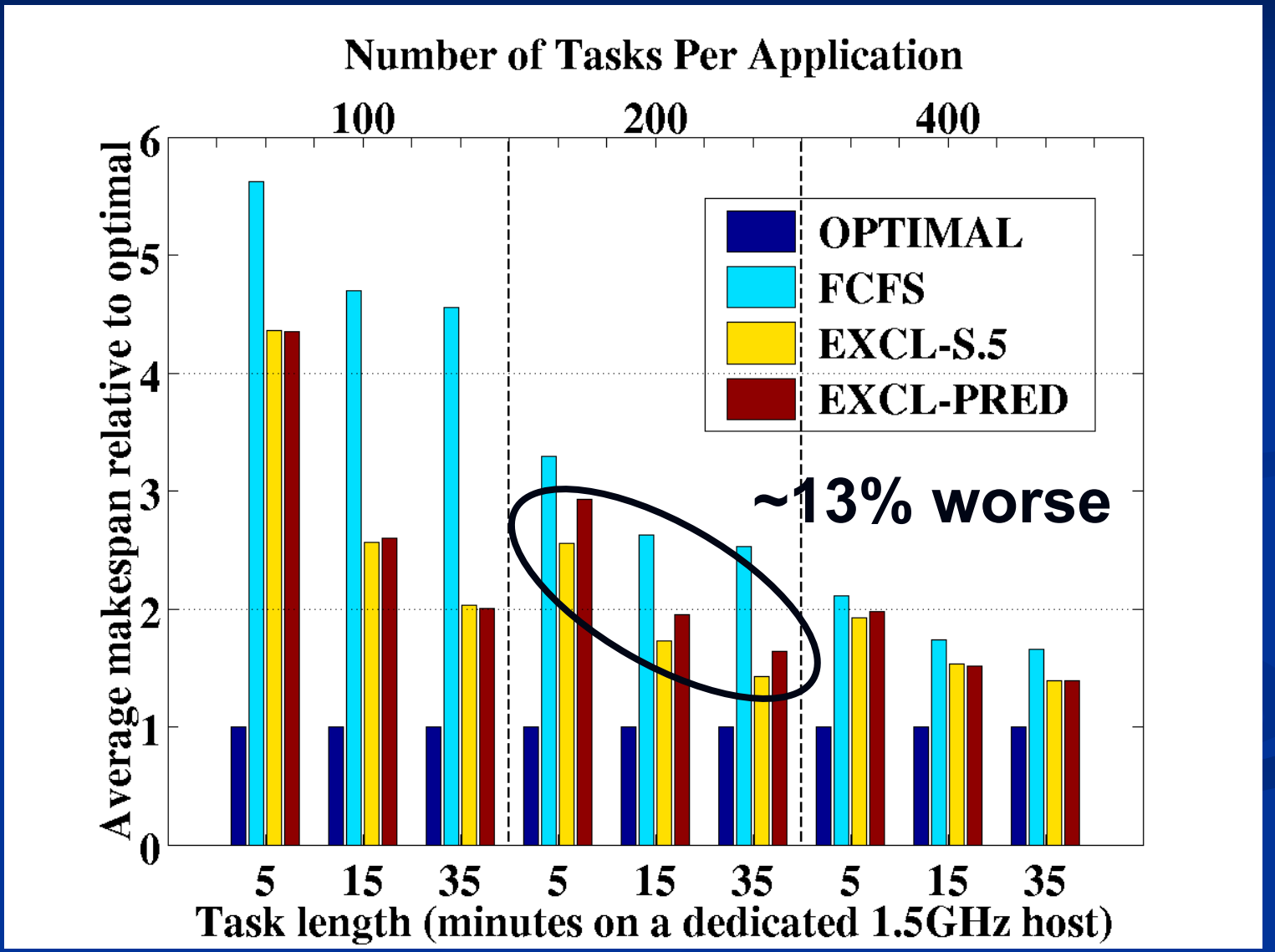
    T is number of tasks

    s is task size in operations
- Compared optimal time with predicted
  - Average error over 1400 experiments is 7.0% with a maximum of 10%

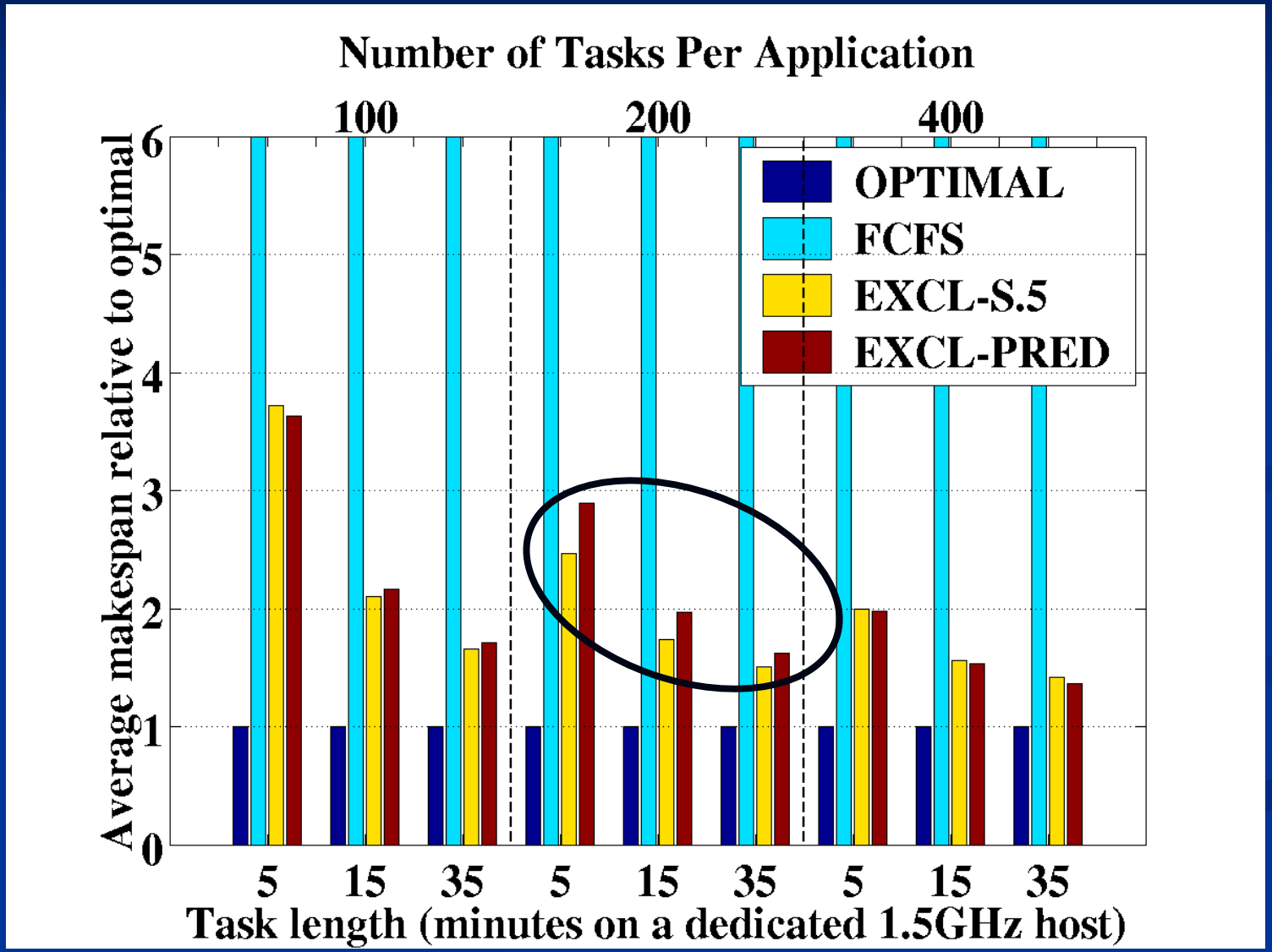
Resource Exclusion Using Predicted Makespans

- EXCL-PRED
  - Compute predicted makespan
  - Exclude hosts that cannot complete task according to its clock rate by predicted makespan
  - After every N tasks are completed, repeat

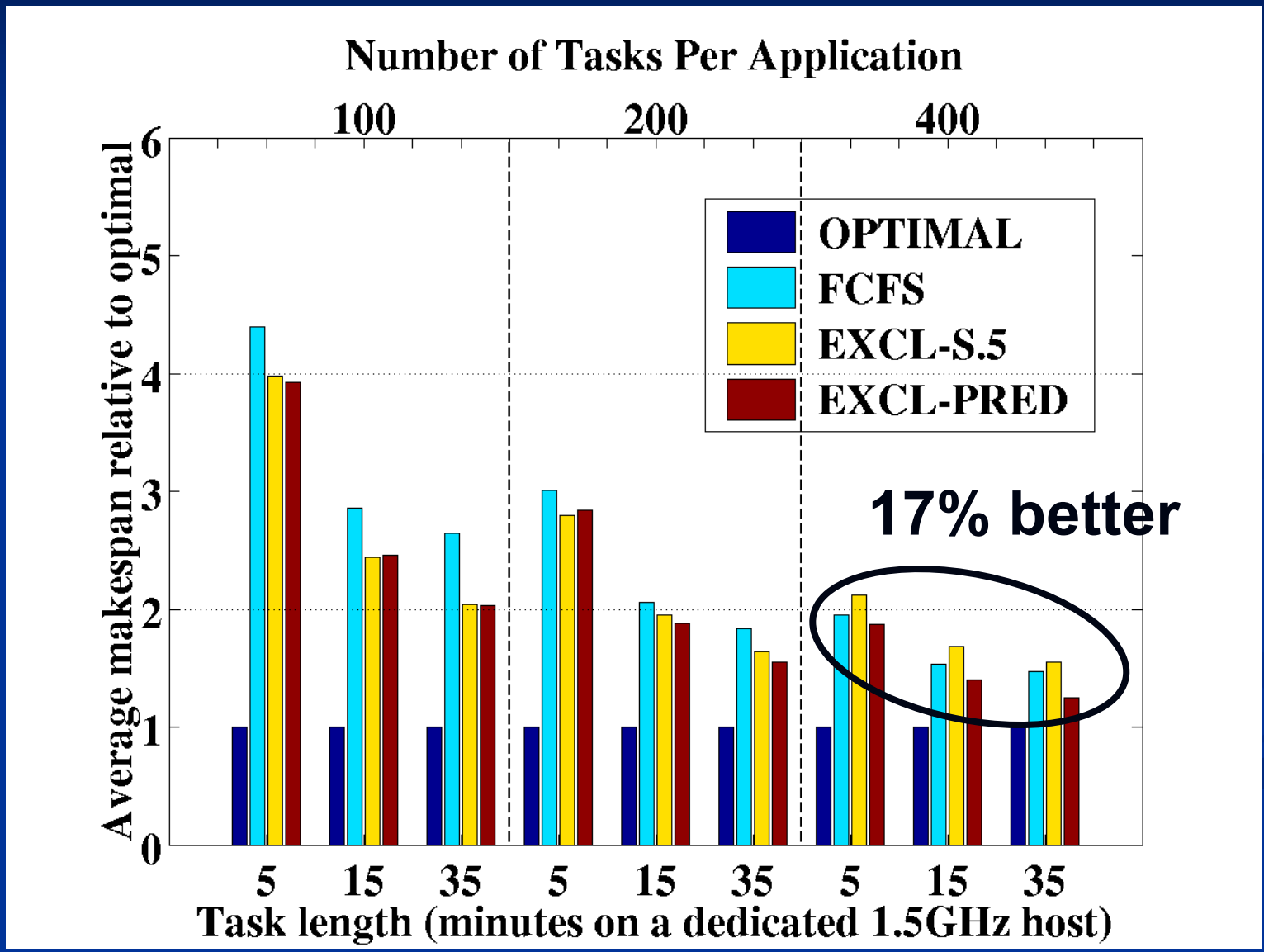
Performance on SDSC grid



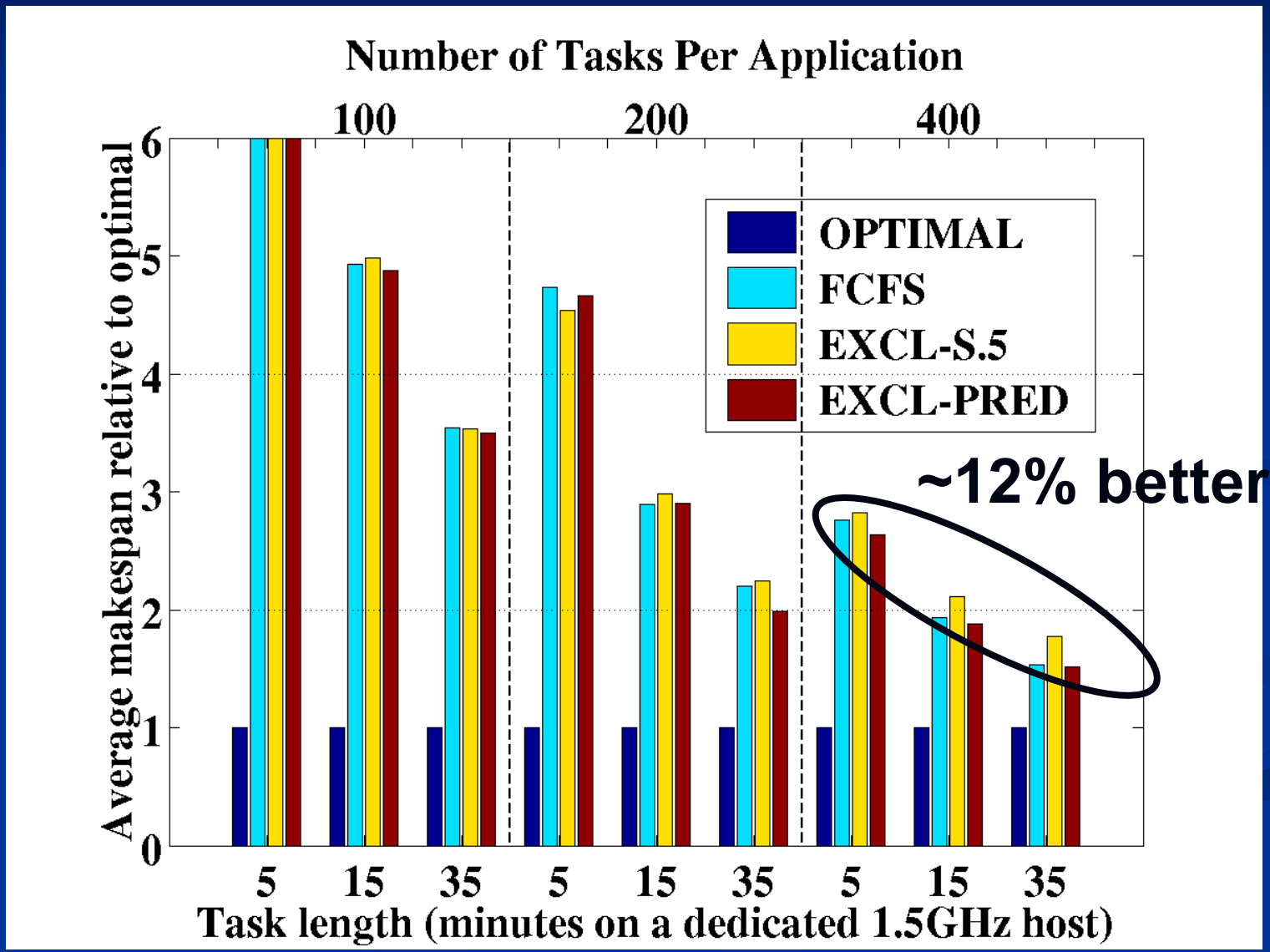
Performance on GIMPS grid



Performance on LRI-WISC Grid



Performance on Homogeneous Grid



Conclusions

- Using a makespan prediction can prevent unnecessary exclusion of useful resources
- Method is susceptible to fluctuation in CPU load
  - Especially with slow hosts in platforms with left-heavy clock rate distributions
- Advantage of using EXCL-PRED clearer when using replication

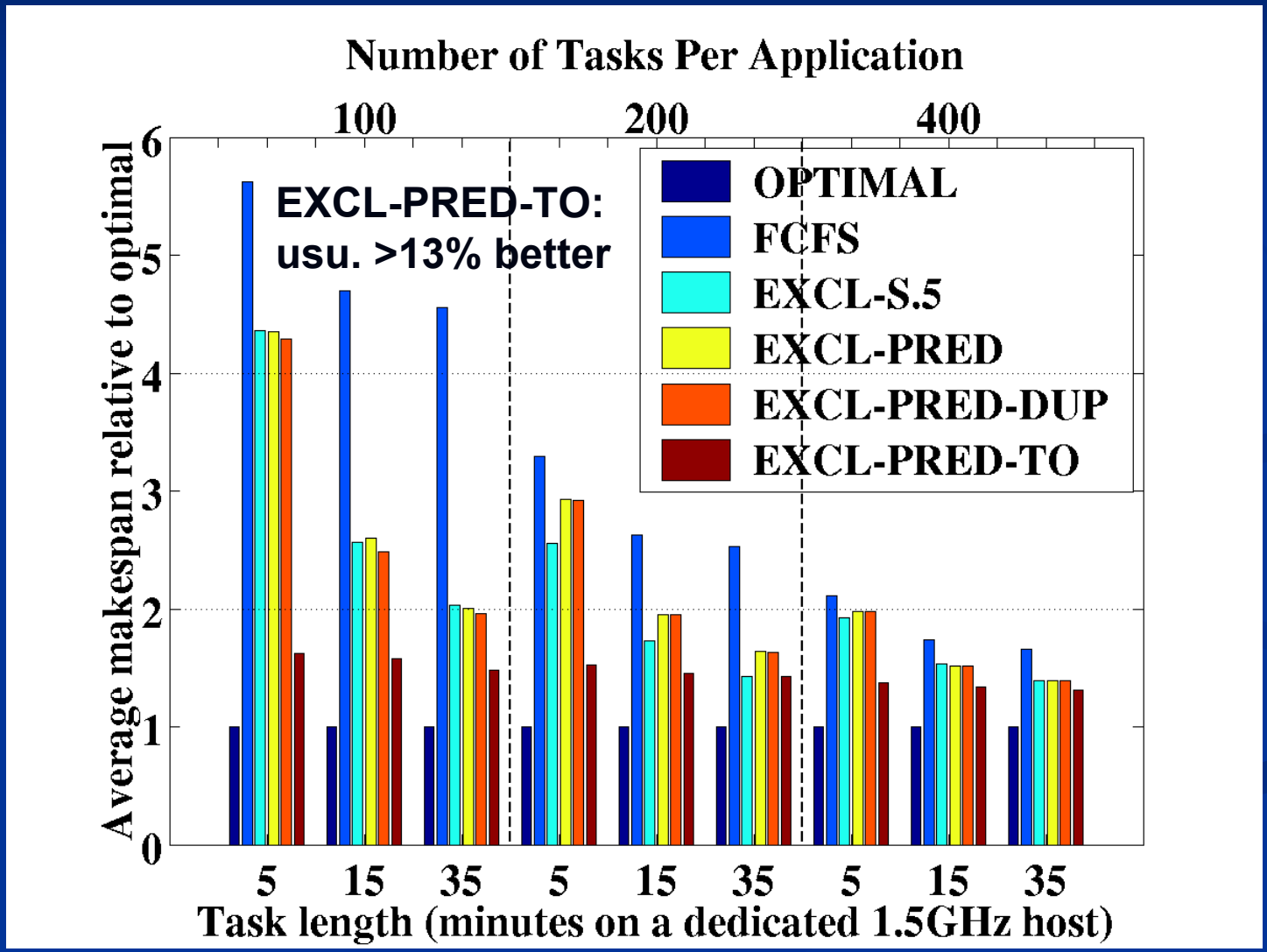
Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

Task Replication

- Replicate to deal with task failures and unpredictably slow hosts
- EXCL-PRED-DUP
  - Only replicate tasks when number of ready hosts is greater than number of tasks to schedule
- EXCL-PRED-TO
  - Timeout per task occurs if it has not been completed by the predicted makespan

Performance with Replication on SDSC Grid



Conclusions

- Replication can improve application performance by masking failures and load fluctuations
  - EXCL-PRED-TO: factor of 1.7 away from optimal on average and on average 2.15 times better than FCFS
- EXCL-PRED-TO performs ~1.5 times better than EXCL-PRED-DUP for the same amount of waste

Outline

- Simulation Method
- Resource prioritization
- Resource exclusion
  - Using a fixed threshold
  - Via makespan prediction
- Task Replication
- Summary

Summary

- Prioritization alone is usually not beneficial
- Using clock rates is an effective criterion for finding good resources
- Excluding resources can improve performance
  - Using a fixed threshold can eliminate useful resources
  - Using makespan predictions can reduce needless resource exclusion but is susceptible to load fluctuations
- Resource exclusion using makespan predictions and replication results in makespans within a factor of 1.7 of optimal and on average 2.15 times better than FCFS

Current and Future Work

- Implement heuristics in Xtremweb software
  - Most desktop grid schedulers are geared towards high-throughput applications
    - XtremWeb, BOINC, Entropia
- Evaluate heuristics with task checkpointing and process migration enabled