Energy-Constrained Dynamic Resource Allocation in a Heterogeneous Computing Environment

B. Dalton Young[1], Jonathan Apodaca[2], Luis Diego Briceno[1], Jay Smith[1,3], Sudeep Pasricha[1,2], Anthony A. Maciejewski[1], Howard Jay Siegel[1,2], Bhavesh Khemka[1], Shirish Bahirat[1], Adrian Ramirez[1], and Yong Zou[1]

Department of Electrical and Computer Engineering[1]
Department of Computer Science[2]
Colorado State University
Fort Collins, Colorado, USA
Dalton.Young@ColoState.edu
DigitalGlobe[3]
Longmont, Colorado, USA

09/12/11

Problem

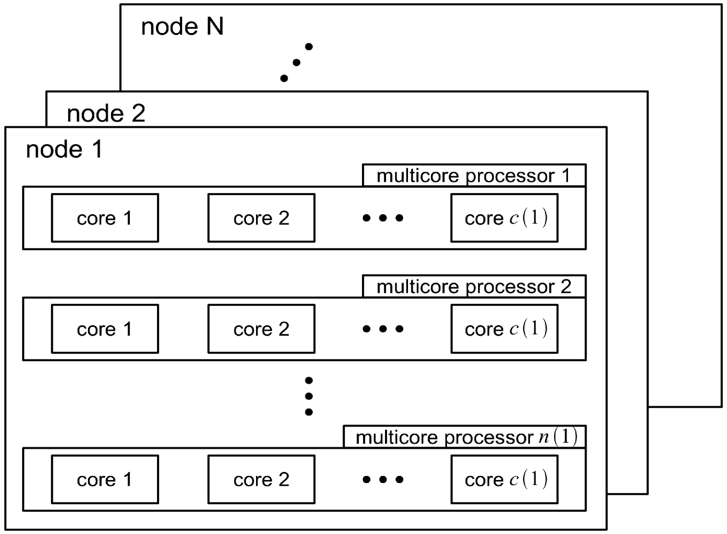
- dynamic resource allocation
- independent tasks with individual deadlines
- goal: complete as many tasks as possible by their individual deadlines
- constraint: total energy consumption
- simulation study

Contributions

- develop model of robustness for our environment
- adapt two existing heuristics
- create a novel heuristic
- demonstrate utility of generalized filter mechanisms

System Model

- multi-core heterogeneous system
  - performance varies between processors
- dynamic, immediate-mode scheduler
  - each task scheduled when it arrives
- P-states from ACPI standard model power/performance tradeoff
- system scheduler controls P-state transitions
- a task cannot be stopped once started

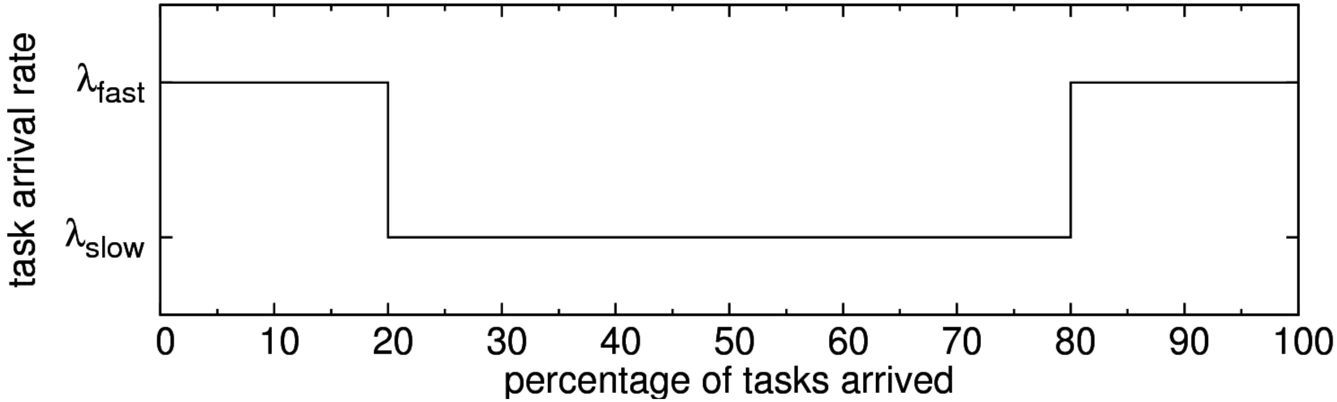


Workload

- collection of known task types
- task type execution time represented by a probability mass function (pmf)
  - found from historical data, experiments, etc. (Li et al., JPDC 1997)
- pmf is scaled to represent execution time in different P-states
- a per-core average power consumption is used for each P-state
- power consumption values generated based on work by Lee and Zomaya (IEEE TPDS 2011)
  - similar to AMD datasheet thermal design power values

Arrival Rate

- bursty arrival rate
- task arrivals modeled as Poisson process
- perfectly subscribed: A reasonable heuristic will finish all tasks on time under the energy constraint with no slack time and no energy remaining.
  - oversubscribed: tasks arrive at a faster rate (λ_{fast})
  - undersubscribed: tasks arrive at a slower rate (λ_{slow})
- slightly undersubscribed on average
- arrival rate structure impacts result



Robustness Questions

- three robustness questions:
  - 1. What makes the system robust?
    - completes tasks by their deadlines
  - 2. What uncertainties are the system robust against?
    - uncertainty in execution time
  - 3. How is robustness quantified?
    - expected value of on-time completions

Calculating Robustness

- expected value of on-time completions
  - from work by Smith et al. (PDPTA 2010)
- when a task arrives, change in robustness is at most 1.0

Heuristics

- used to assign each task when it arrives
  - optimize number of tasks completed under constraint on the total energy consumed
- assignment: mapping of task to a node, multi-core processor, core, and P-state
- can use filters to add energy- and robustness-awareness
- may leave tasks unassigned

Heuristics: Random

- randomly assign task
- used for comparison

Heuristics: Shortest Queue

- minimize number of tasks assigned to each core
- tiebreaker: expected execution time

Heuristics: Minimum Expected Completion Time

- minimize task's expected completion time
- completion time: sum of expected task execution times and current time

Heuristics: Lightest Load

- attempt to balance energy and robustness by minimizing a “load” L
- En_{ex} : expected energy consumed
- ΔR : change in robustness

$$L = (1.0 - \Delta R) \times En_{ex}$$

Energy Filter

- filter tracks estimated energy remaining
- restrict potential assignments using energy threshold En_{thresh}
- En_{rem}: estimated energy remaining
- T_{rem} : tasks remaining in the workload
- En_{mul}: multiplier from average queue depth

$$En_{thresh} = En_{mul} * En_{rem} / T_{rem}$$

Robustness Filter

- restrict potential assignments using a robustness change threshold ΔR_{thresh}

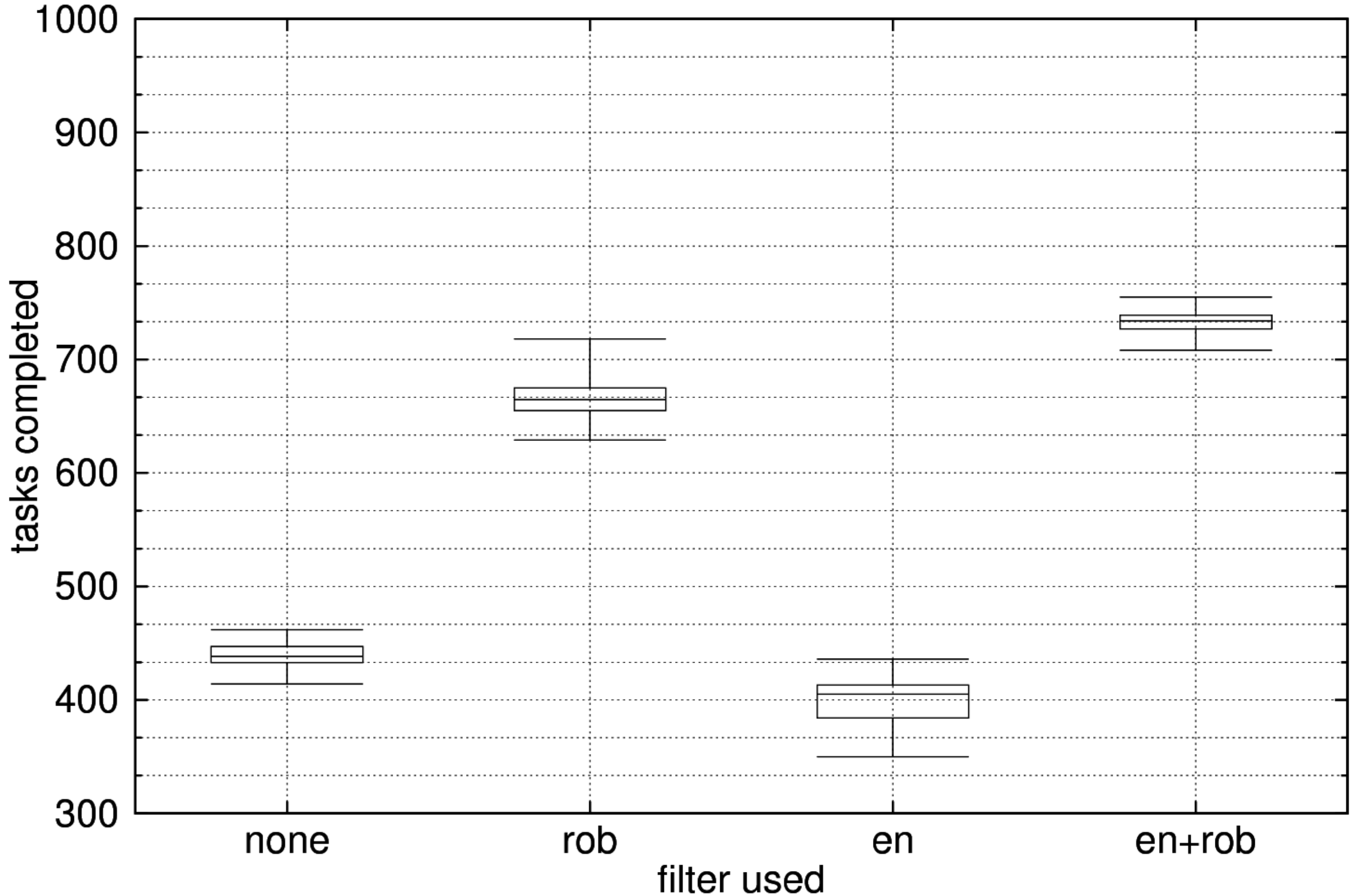
$$\Delta R_{thresh} = 0.50$$

Simulations

- 50 trials, 1000 tasks each trial, 100 task types
- task type pmfs generated using Coefficient of Variation Based method (Ali et al., TJSE 2000)
- energy constraint: product of average task execution time, average power, and number of tasks
- variations between simulation trials:
  - task-type mix
  - task arrival times
  - task execution times
  - task deadlines

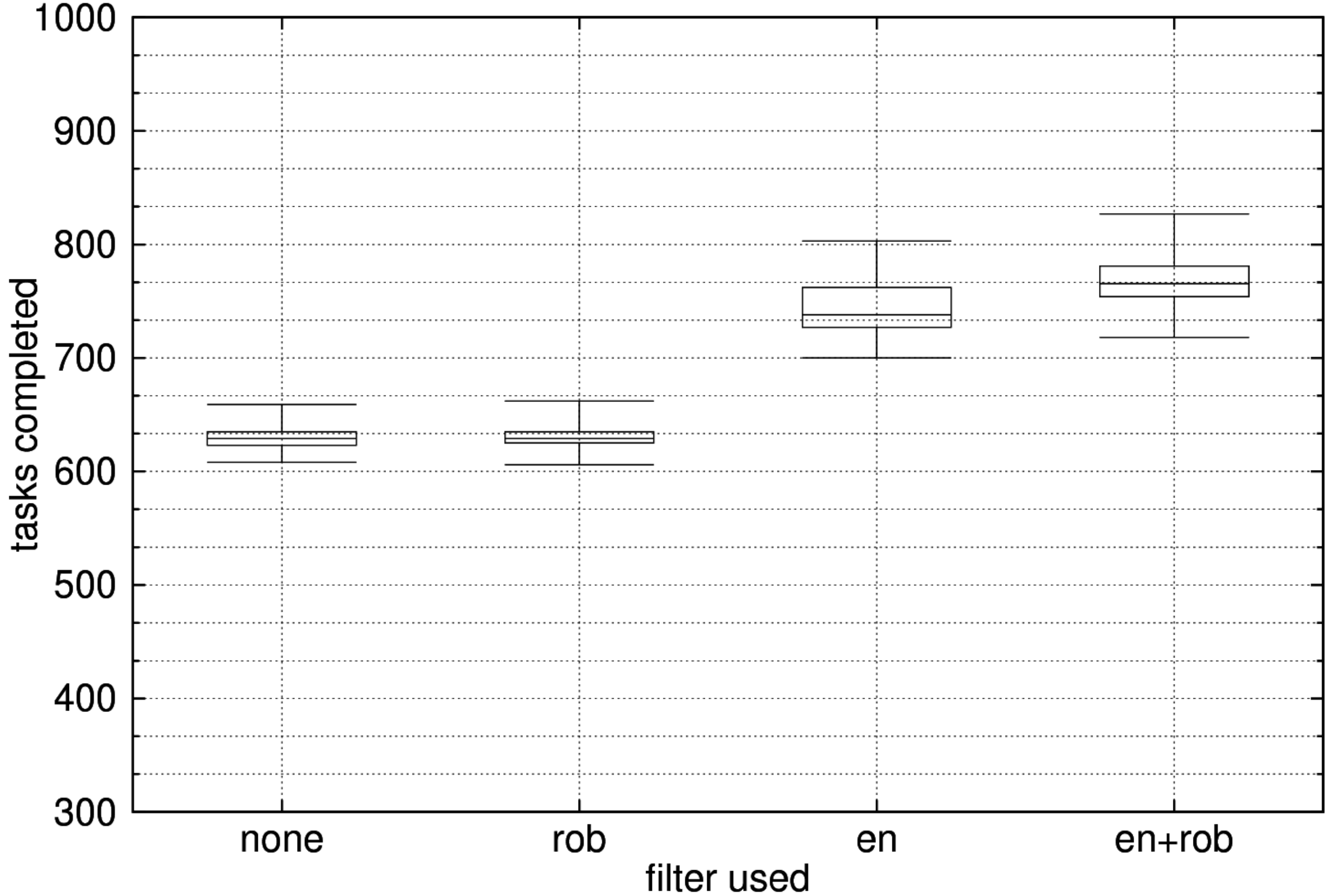
Results: Random

- robustness filter more useful than energy
- combined filtering best (~60 additional completions)



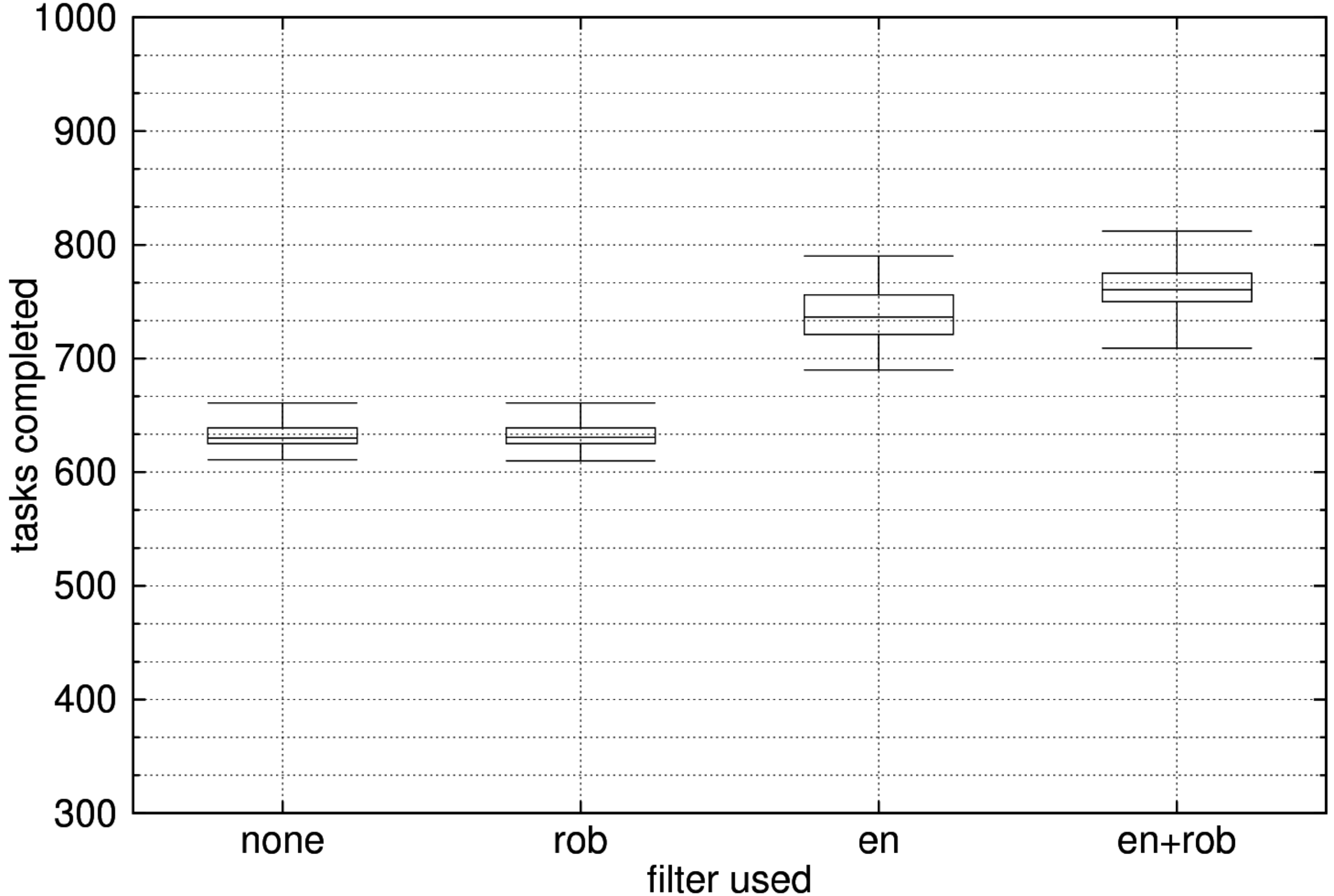
Results: Shortest Queue

- robustness filtering useful with energy filtering
- energy filtering ~100 completions better than no filtering



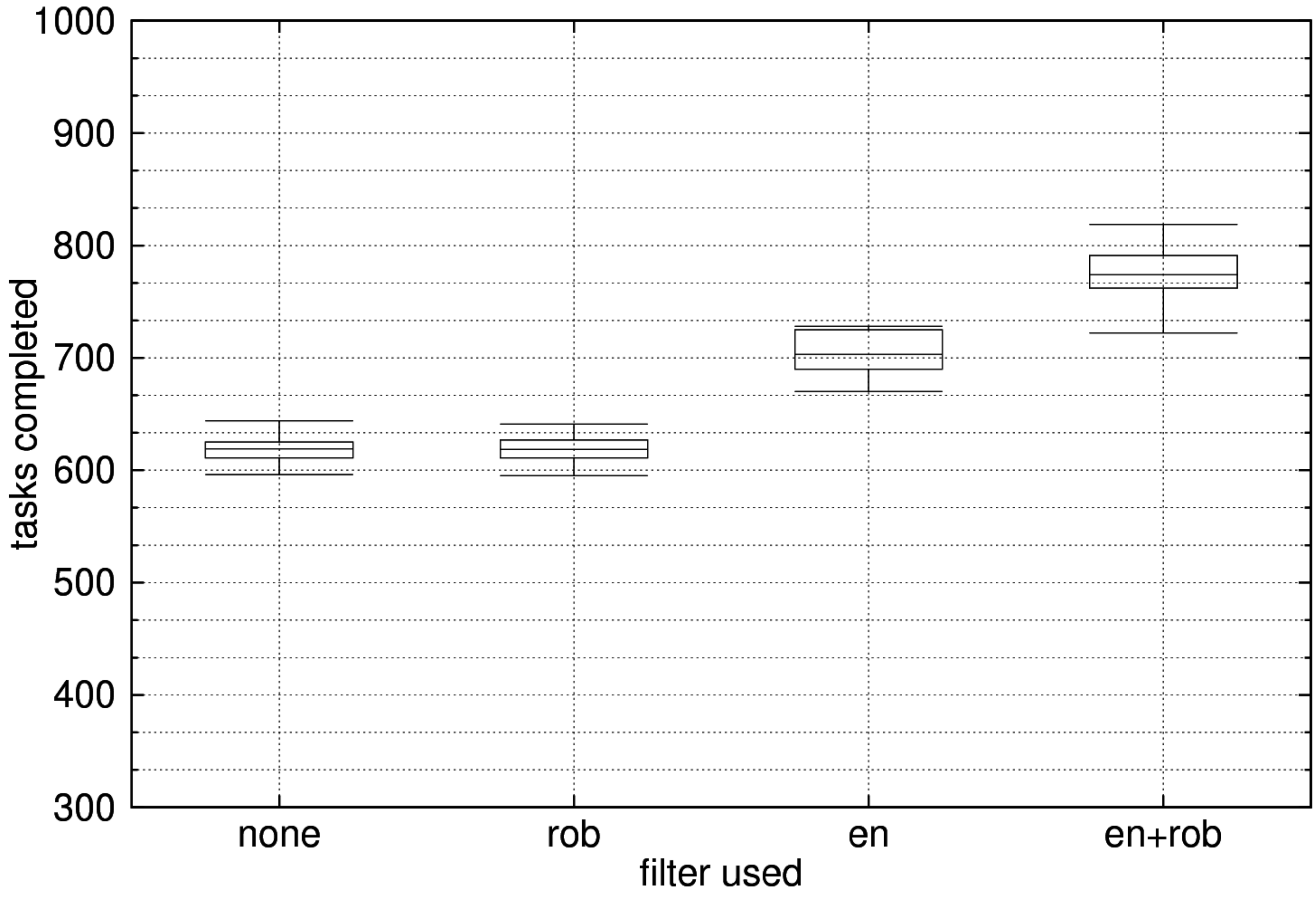
Results: Minimum Expected Completion Time

- robustness filtering useful with energy filtering
- energy filtering ~100 completions better than no filtering



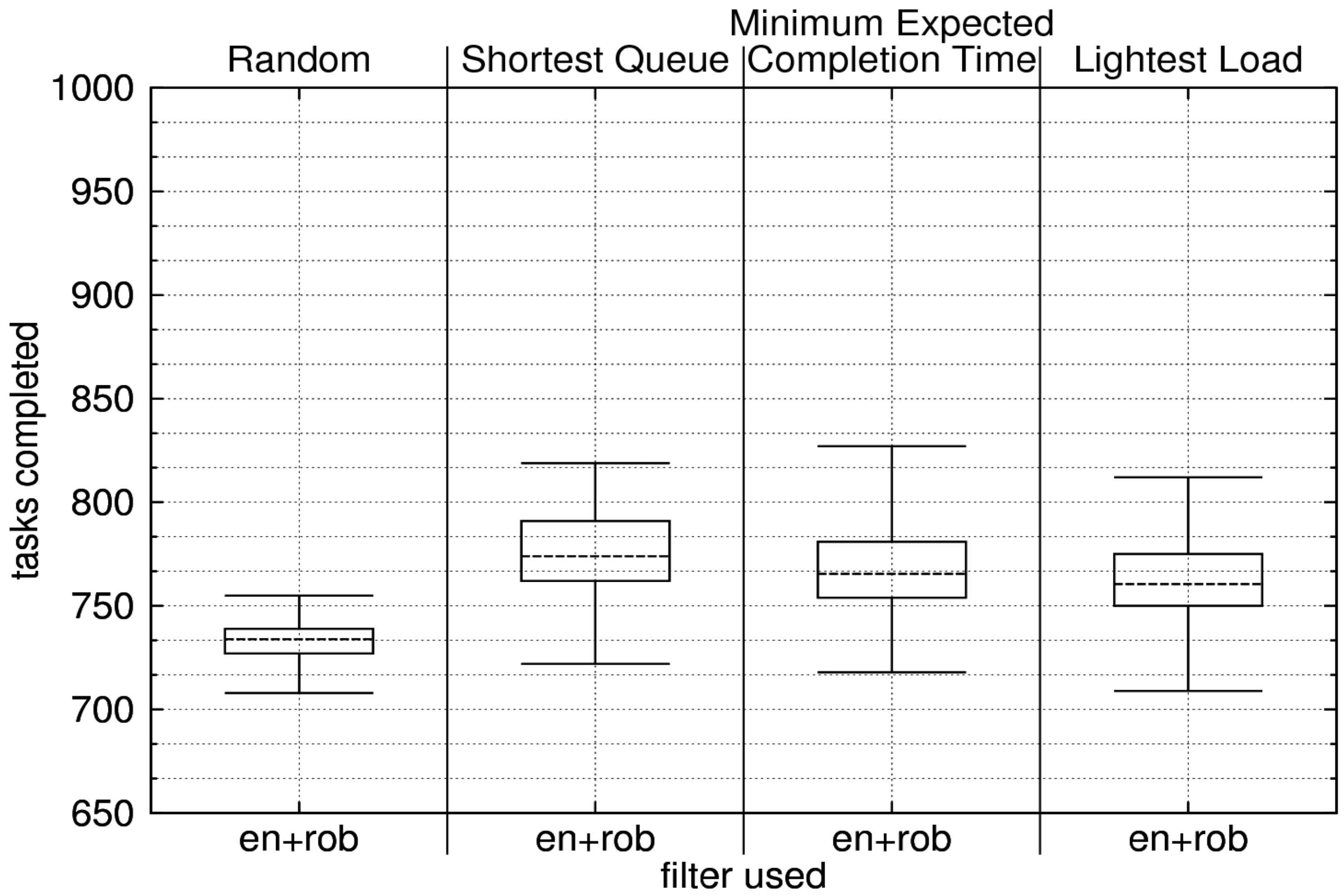
Results: Lightest Load

- robustness filtering useful even though load has robustness
- energy filtering ~90 completions better than no filtering



Results: Best Comparison

- all best results use energy and robustness filtering
- random median within 4% of best value



Conclusions

- filtering mechanisms more important than heuristic
- important to take energy into account
- robustness model useful in conjunction with an energy-aware filter

Future Work: Power

- try more power-saving mechanisms
  - could include ACPI G-states
  - could include turning machines off
- use power distributions instead of averages
- consider non-CPU power (memory, disks, etc)

Future Work: System Model and Simulations

- task cancellation to mitigate bad assignments
- tasks with priorities
- different arrival rates and structures
- stop tasks as soon as deadline missed

Questions?