Energy-Aware Robust Resource Management for Parallel Computing Systems

H. J. Siegel
Abell Endowed Chair
Distinguished Professor of
Electrical and Computer Engineering
and Professor of Computer Science

Colorado State University
Fort Collins, Colorado, USA

Outline

- stochastic model for resource allocation
- static resource allocation with energy minimization
- dynamic resource allocation with energy constraint
- conclusions

Heterogeneous Parallel Computing System

- interconnected set of different types of machines with varied computational capabilities
- workload of tasks with different computational requirements
- each task may perform differently on each machine
  - furthermore: machine A can be better than machine B for task 1 but not for task 2

Resource Management

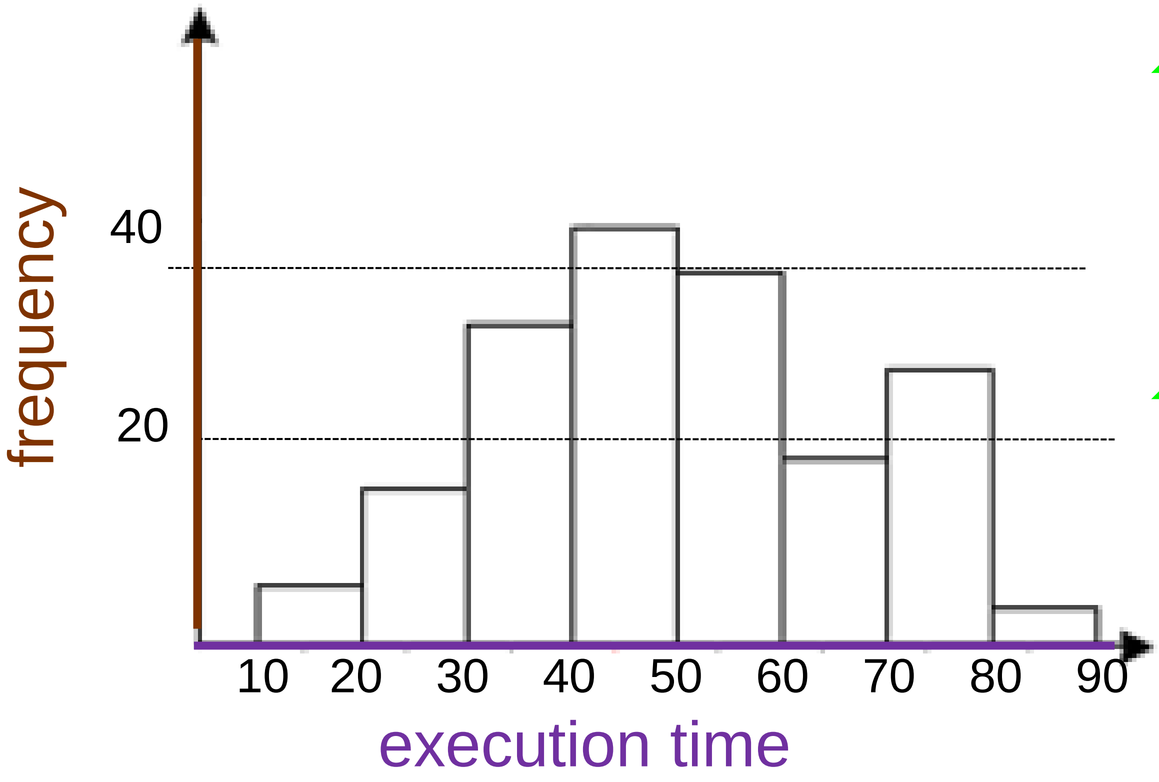
- assign tasks to machines
  - optimize some performance measure
  - possibly meet system constraint
- in general, known **NP-complete** problem
  - cannot find optimal solution in reasonable time
  - ex.: 5 machines and 30 tasks
    $\rightarrow$ 5^{30} possible assignments
  - if it only took 1 nanosecond to evaluate each assignment
    - 5^{30} nanoseconds > 1,000 years!
  - use **heuristics** to find near-optimal solutions

Stochastic Model for Robustness

- reference
  - “Stochastic Robustness Metric and its Use for Static Resource Allocations”
  - by Shestak, Smith, Maciejewski, and Siegel
  - *Journal of Parallel and Distributed Computing*
  - August 2008, Vol. 68, No. 8, pp. 1157-1173

Modeling Uncertain Task Execution Times

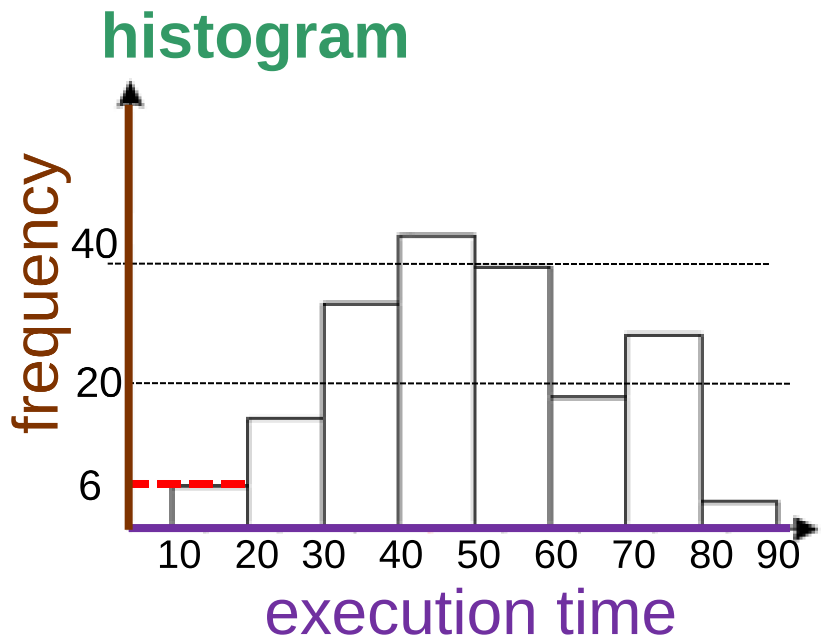
- execution of a given task on a given machine is data dependent
- collect in a **histogram** a history samples of
  - execution time of a given task on a given machine
  - over different representative data sets

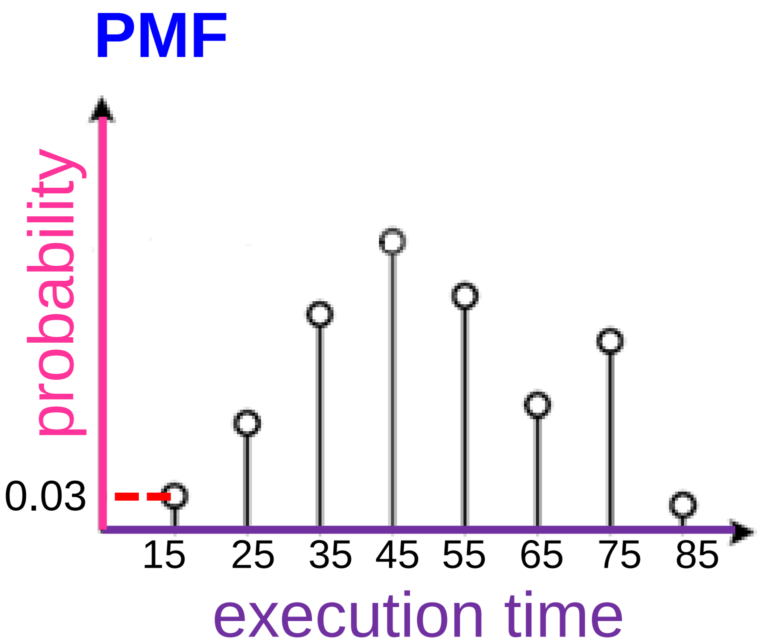


- x-axis: execution time within 10 second interval bins
- y-axis: frequency = height of bar for a given interval

Generating a PMF from a Histogram

- a probability mass function (PMF) can be generated using a histogram
- convert the frequency to a probability to create PMF
  - probability = frequency/total # samples
- example: probability of value from 10 to 19 = 6/200 = 3%

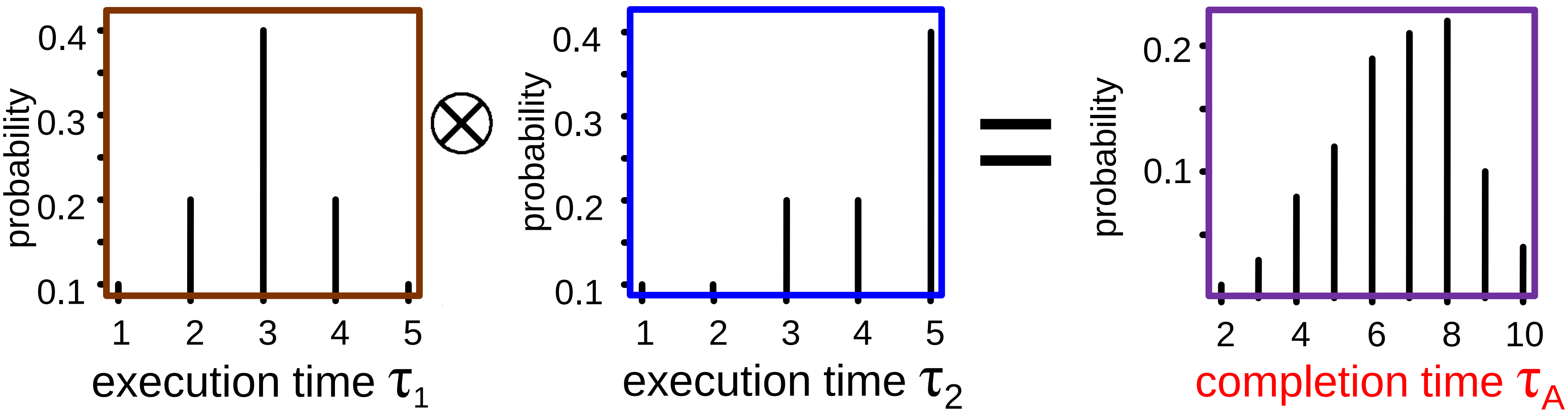




PMF for Completion Time of Machine

- assume task 1 and task 2 only tasks assigned to machine A
  - can find completion time PMF for machine A to do both tasks
  - if tasks independent, it is the "discrete convolution" (combination) of the execution time PMFs for the two tasks

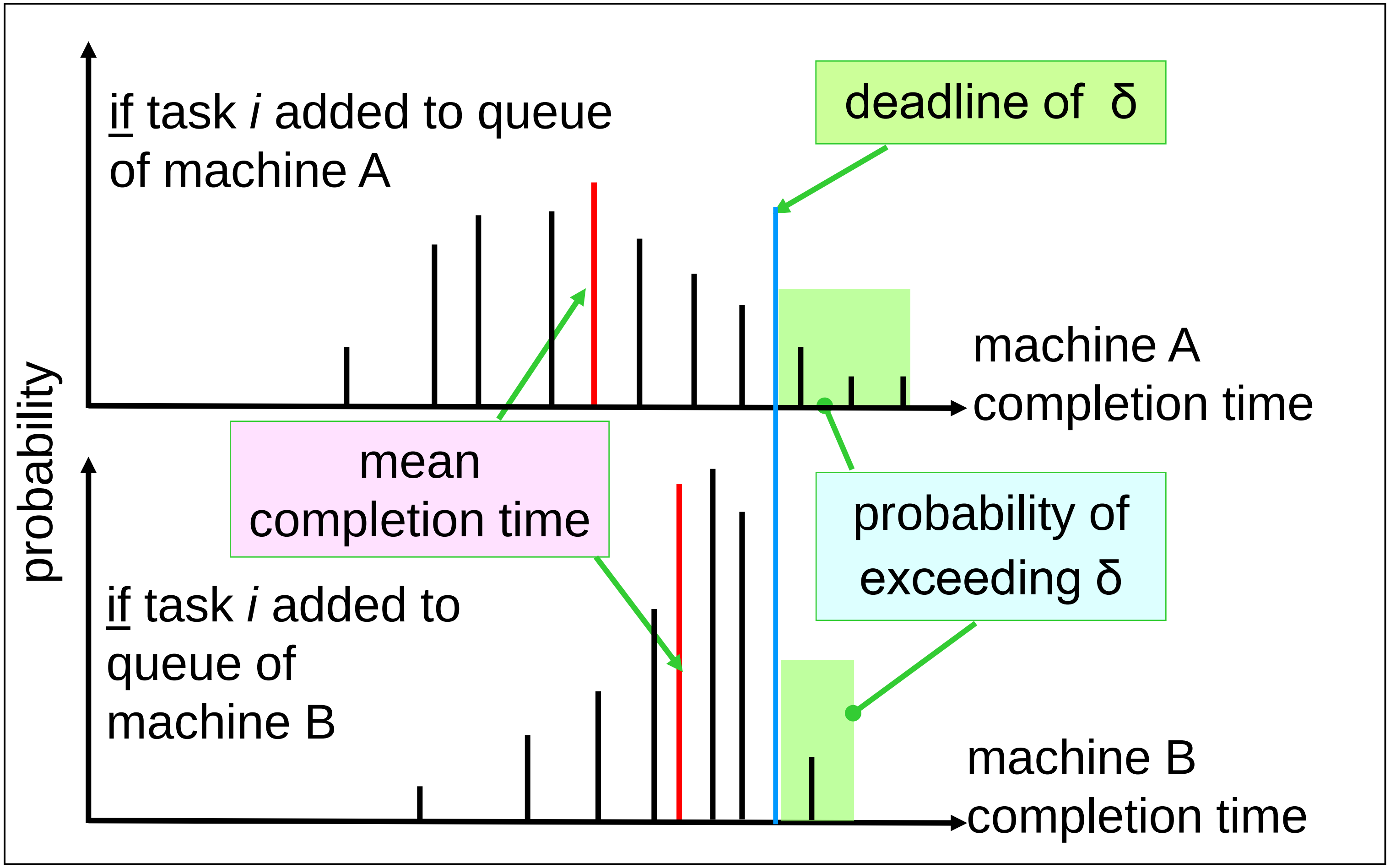
PMF for t_1 on machine A | PMF for t_2 on machine A | PMF for completion time of machine A



- $p(\tau_A = k) = \sum_{\tau_1 + \tau_2 = k} (p(\tau_1) \cdot p(\tau_2))$

Example of Use of Stochastic Model in Allocation

- PMFs for machine completion time based on
  - (1) PMFs for tasks already assigned to that machine, and
  - (2) PMF for task *i* – which <u>may</u> be assigned to that machine



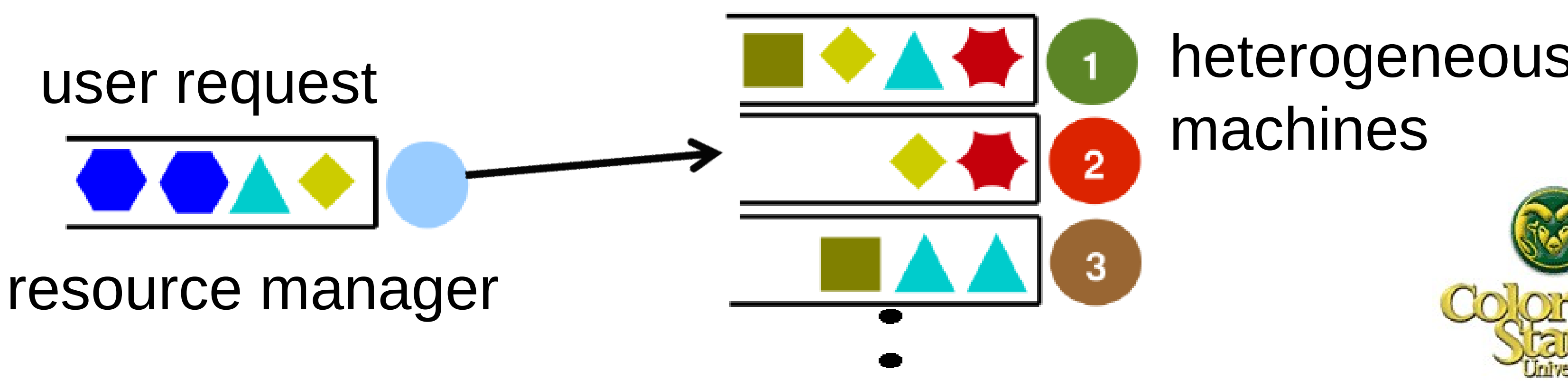
- assign task *i* to machine A or B?
- mean → A
- sum of heights of pulses > deadline → B

Static Resource Allocation

- determine allocation of set of tasks to machines off-line
- know in advance which tasks are to be executed during a given interval of time (e.g., the next day)
- "bag-of-tasks"
- uses
  - planning future work in production environment
    - e.g., resource manager plans when and where tasks will execute the next day
  - predictive "what-if" studies
    - e.g., system administrator wants to quantify the benefit of adding more machines to the network
  - post-mortem analysis of a dynamic heuristic
    - e.g., static allocation based on trace to compare performance to dynamic results

Dynamic Resource Allocation

- tasks assigned to machines as they arrive (on-line)
- tasks are from a known set (e.g., Digital Globe, NCAR, ORNL)
- do <u>not</u> know in advance
  - which tasks (from the known set) will need to be executed
  - when tasks will arrive
  - what data sets will be processed
- set of machines in the computing system can change
- can use feedback about status of machines
- because done as tasks arrive,
  must execute faster than static heuristics



Outline

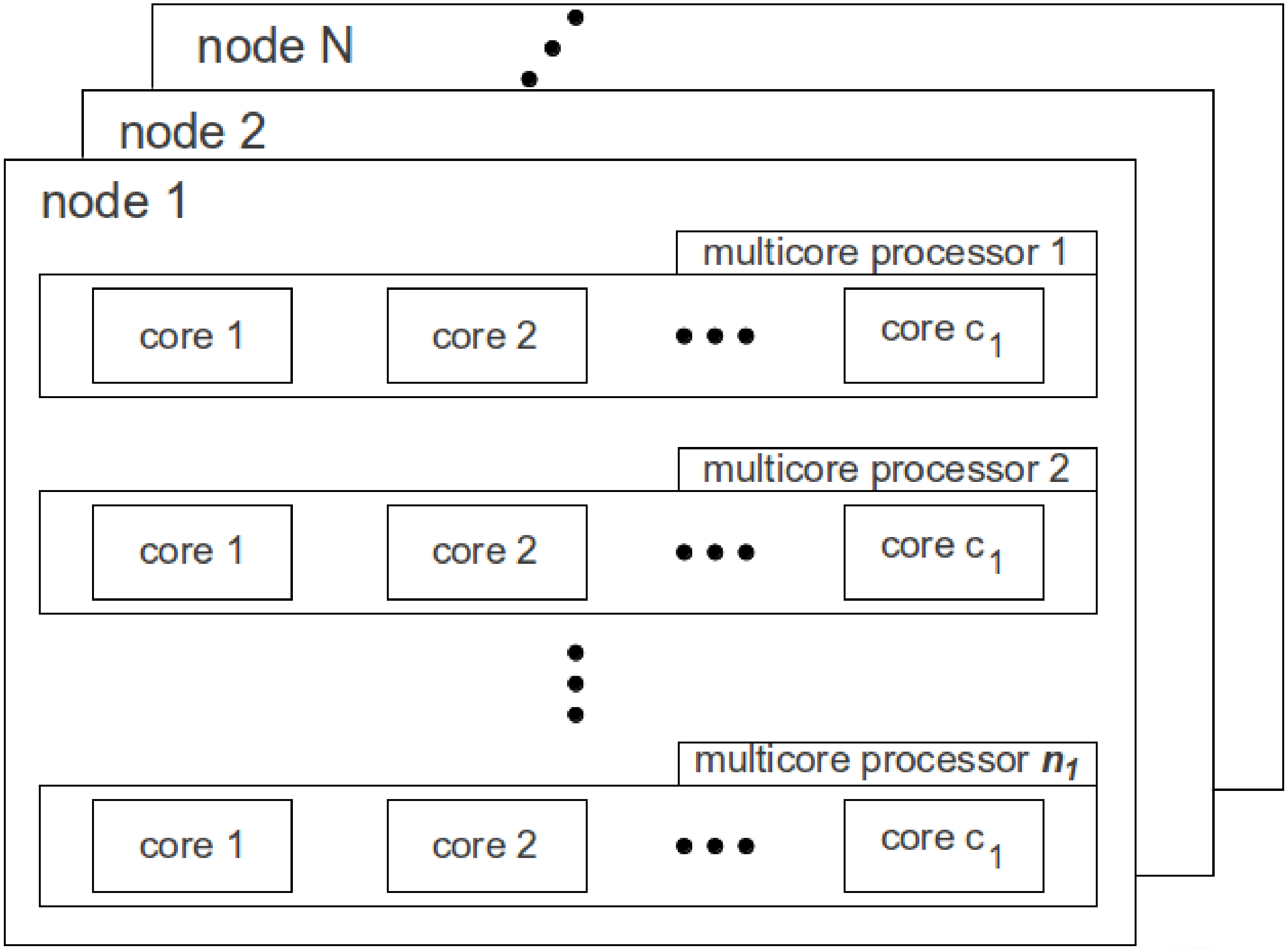
- stochastic model for resource allocation
- static resource allocation with energy minimization
- dynamic resource allocation with energy constraint
- conclusions

Static Heuristics with Energy Minimization

- reference
  - "Stochastically Robust Static Resource Allocation for Energy Minimization with a Makespan Constraint in a Heterogeneous Computing Environment"
  - by Apodaca, Young, Briceño, Smith, Pasricha, Maciejewski, Siegel, Bahirat, Khemka, Ramirez, and Zou
  - *9th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA '11)*
  - December 2011

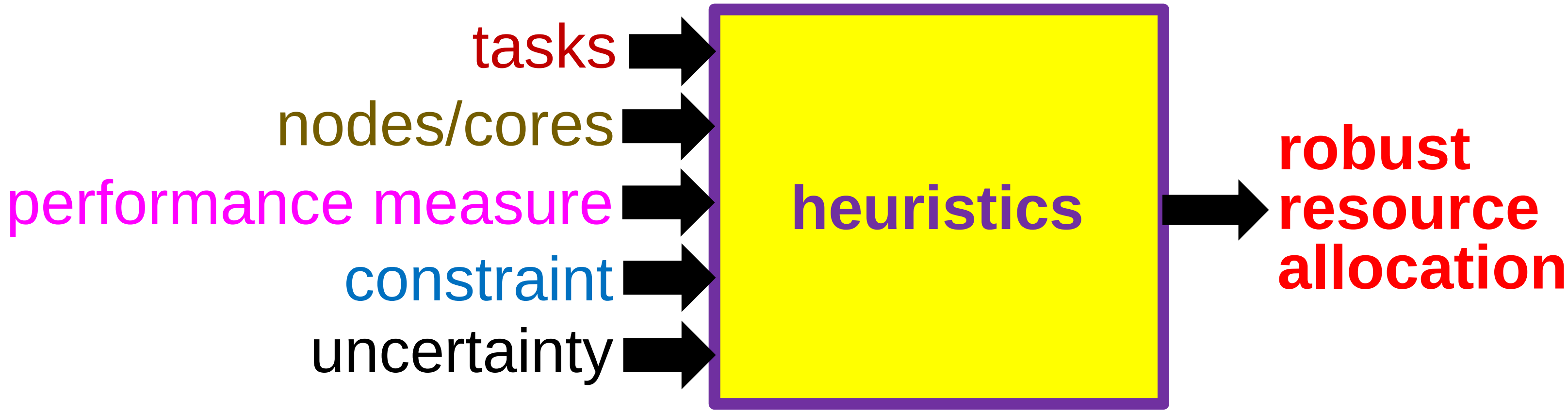
Architecture Model

- N heterogeneous compute nodes
- each compute node i has n_i homogeneous multicore processors, $1 \le n_i \le 4$
- each multicore processor j in compute node i has c_i homogeneous cores, $1 \le c_i \le 4$



Problem Statement for Static Study

- known collection of independent tasks
- common deadline δ to complete all tasks
- uncertainty in execution time of given task on given core type due to data dependencies is represented as PMF
- energy used is concern because of costs
- goal: design robust resource management techniques that
  - minimize expected energy used (performance measure)
  - constraint on probability of finishing by deadline (robustness)



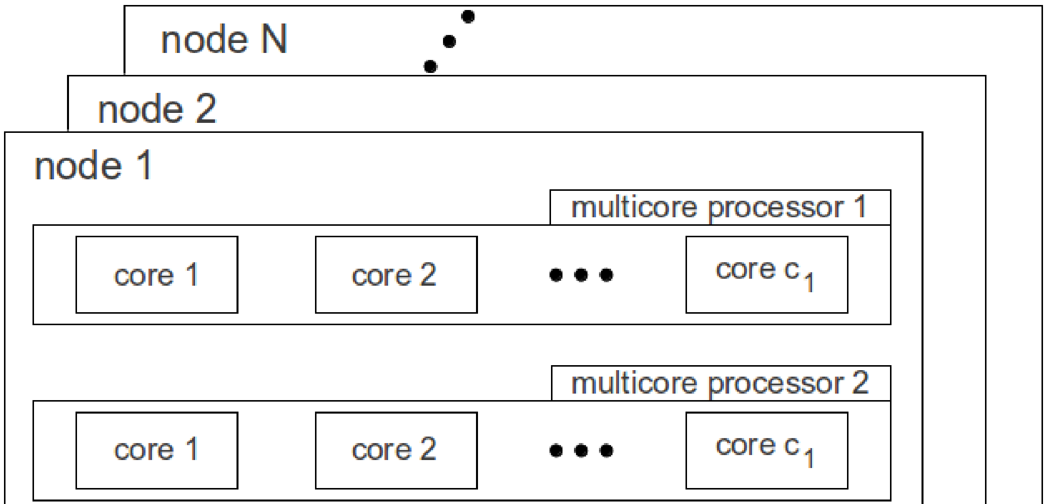
Robustness Definition

- term "robustness" usually used without explicit definition

- THE THREE ROBUSTNESS QUESTIONS
  1. what behavior makes the system robust?
     - finishing all tasks by the common deadline δ
  2. what uncertainty is the system robust against?
     - each task's execution times may vary substantially based on input data
  3. how do we quantify robustness?
     - the probability that a given resource allocation will complete all tasks by the common deadline δ

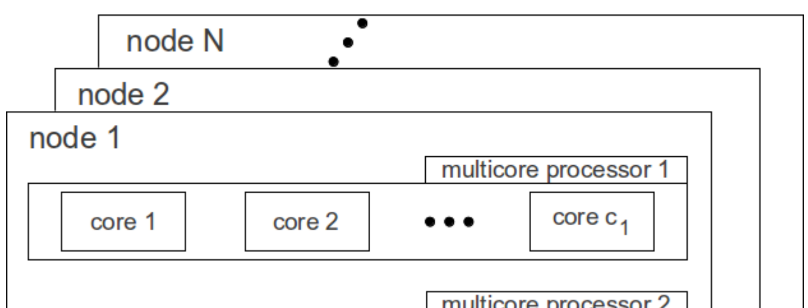
Energy Model – Hierarchy

- nodes shut off when all internal multicore processors are idle
  - when one or more internal multicore processors are on, however, a node incurs power overhead (e.g., for disks, fans)
- multicore processors shut off when all internal cores are idle
  - when one or more internal cores are on, however, a multicore processor incurs power overhead (e.g., for L3 cache)
- each core executes continuous sequence of tasks
  - shut down core/processor/node ASAP



Energy Model – DVFS

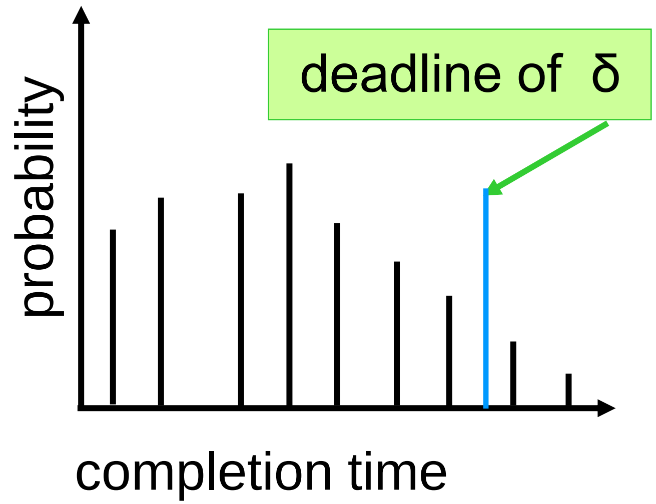
- each core uses Dynamic Voltage and Frequency Scaling (DVFS)
- five P-states (performance states)
  - P0 highest power to P4 lowest power
  - higher power consumption → faster execution
  - typically – lower power P-state → less energy but more time
    - depends on ratio of overhead energy to CPU energy
    - type of task: memory-intensive, CPU-intensive
- execution time PMF for each task - each core type - each P-state
- for each P-state for each core type, average scalar value for power consumption (energy per second)
- cores can switch states independently - negligible overhead



Formal Definition of Robustness for Static Study

- given a resource allocation (including P-state assignments)
  - let D_{ijk} be the finishing time distribution PMF for all tasks assigned to core k in multicore processor j in compute node i
  - let $p(D_{ijk}, \delta)$ be probability of finishing before δ given D_{ijk}
    - sum of pulses < δ in PMF
- overall system robustness ψ
  - probability of all tasks finishing by δ

$$\psi = \prod_{1 \le i \le N} \prod_{1 \le j \le n_i} \prod_{1 \le k \le c_i} p\left(D_{ijk}, \delta\right)$$



Heuristics for Static Study

- recall - goal: design robust resource management techniques
  - minimize expected energy used (performance measure)
  - constraint on probability of finishing by deadline (robustness)
- the robustness constraint is ***R***%
  - this could be specified by the system administrator
  - simulation study: we use robustness constraint to be 90%
- heuristics from the paper
  - Min-Min
  - Genetic Algorithm (GA)
  - Tree Search
  - Tabu

Genetic Algorithm (GA) – Chromosome

- chromosome structure – represents possible solution (allocation)
  - number of genes (length) = number of tasks to be mapped
  - t^{th} entry is a four-tuple (i, j, k, π)
  - denotes mapping **task *t*** to **node *i***, **multicore processor *j***, **core *k***, in **P-state *π***
  - order of task execution within a core does not matter

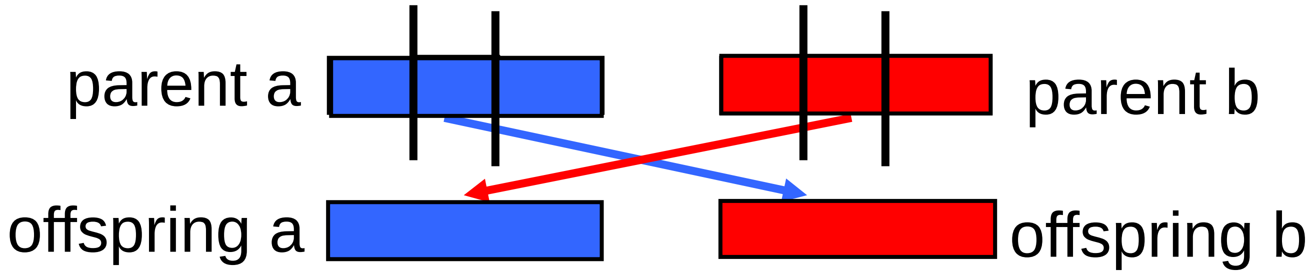
task	1	2	3	4	...
node	2	1	2	3	...
processor	4	2	1	1	...
core	2	2	3	1	...
P-state	0	4	2	1	...

Genetic Algorithm (GA) - Population

- fixed size population of chromosomes – collection of solutions
- Genitor-style GA (steady-state GA)
- population ordered by fitness value as follows:
  - chromosomes that meet robustness constraint in increasing order of expected energy (lower better)
  - rest in decreasing order of robustness (higher better)
- initial population generation
  - five seeds based on Min-Min
    - greedy heuristic
    - run for a fixed P-state
    - done 5 times, 1 per P-state
  - rest simple greedy heuristic that meets constraint

GA – Crossover Operation

- randomly select a pair of "parents" for crossover with a probability p_c
- choose two points x & y such that $x < y \leq$ number of tasks
- swap genes in range [x, y] between chromosomes
- generates two offspring



GA – Task-Assignment Mutation Operation

- each chromosome has probability p_{tm} of being mutated
- each gene within selected chromosome has probability p_{tmg} of being mutated
- change assignment to random core, in random P-state

task	1	2	3	4	…
node	2	1	2	3	…
processor	4	2	1	1	…
core	2	2	3	1	…
P-state	0	4	2	1	…

4
3
1
0

GA – P-State Mutation Operation

- each chromosome has probability p_{pm} of being mutated
- each gene within selected chromosome has probability p_{pmg} of being mutated
- change P-state of random task to random P-state

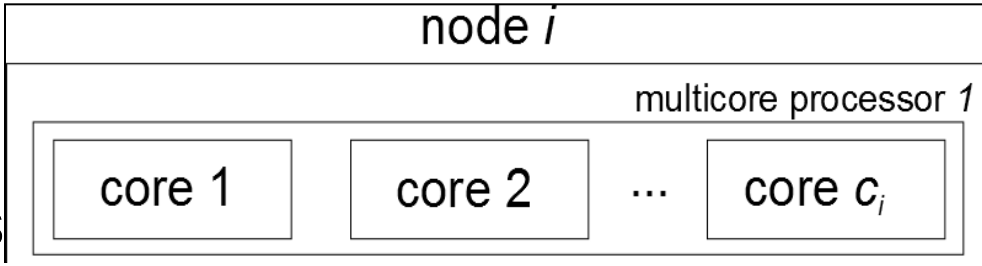
task	1	2	3	4	…
node	2	1	2	3	…
processor	4	2	1	1	…
core	2	2	3	1	…
P-state	0	4	2	1	…

0

GA – Procedure Overview

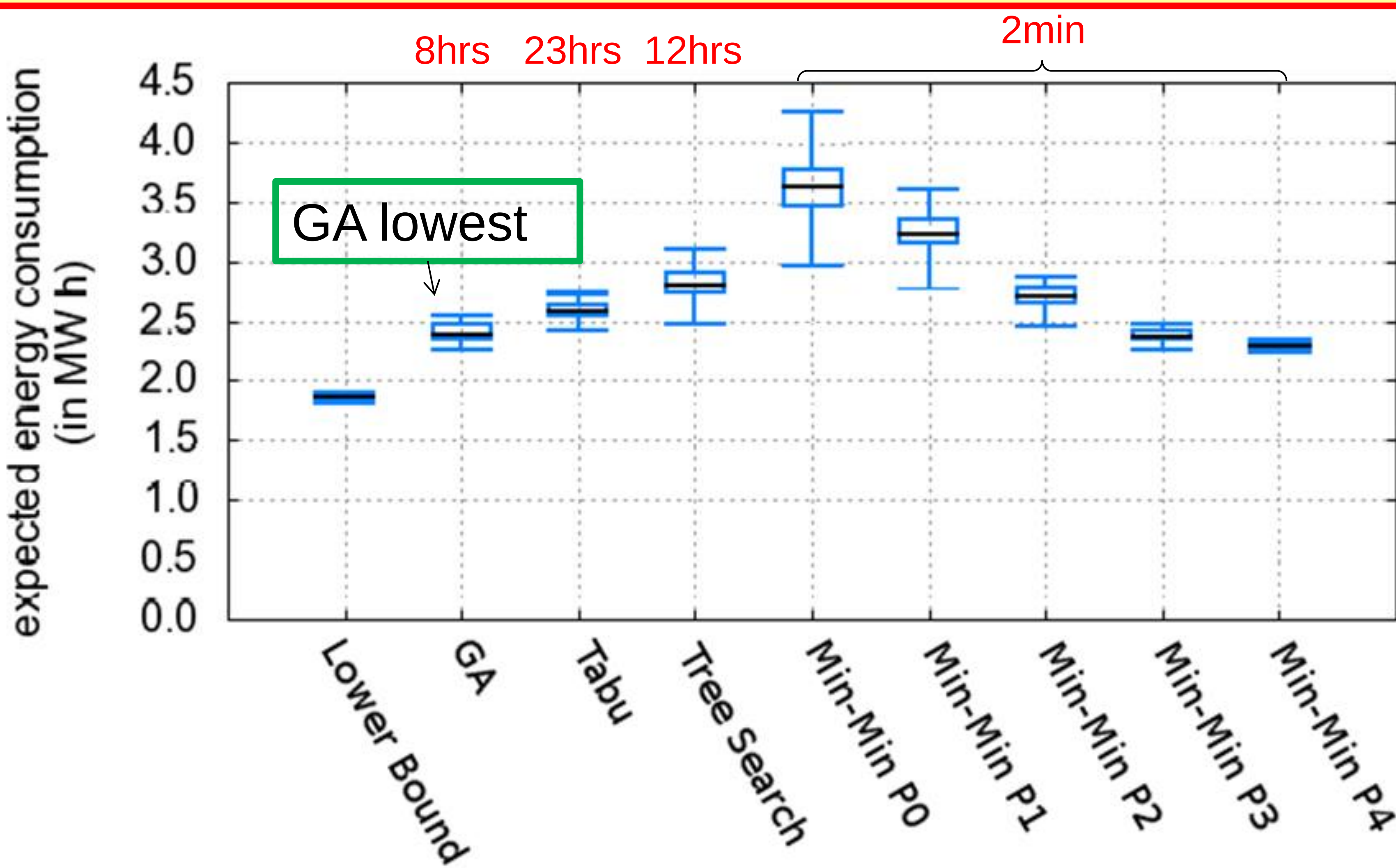
- generate initial population (size denoted *S*)
- repeat for a given number of iterations
  - do *S* times: choose two random chromosomes, and with probability p_c produce two offspring via crossover
    - insert offspring in ordered population and trim to size *S*
  - for each chromosome in population, make offspring via task-assignment mutation with probability p_{tm}
    - insert offspring in ordered population and trim to size *S*
  - for each chromosome in population, make offspring via P-state mutation with probability p_{pm}
    - insert offspring in ordered population and trim to size *S*
- return best chromosome encountered

Simulation Setup for Static Study



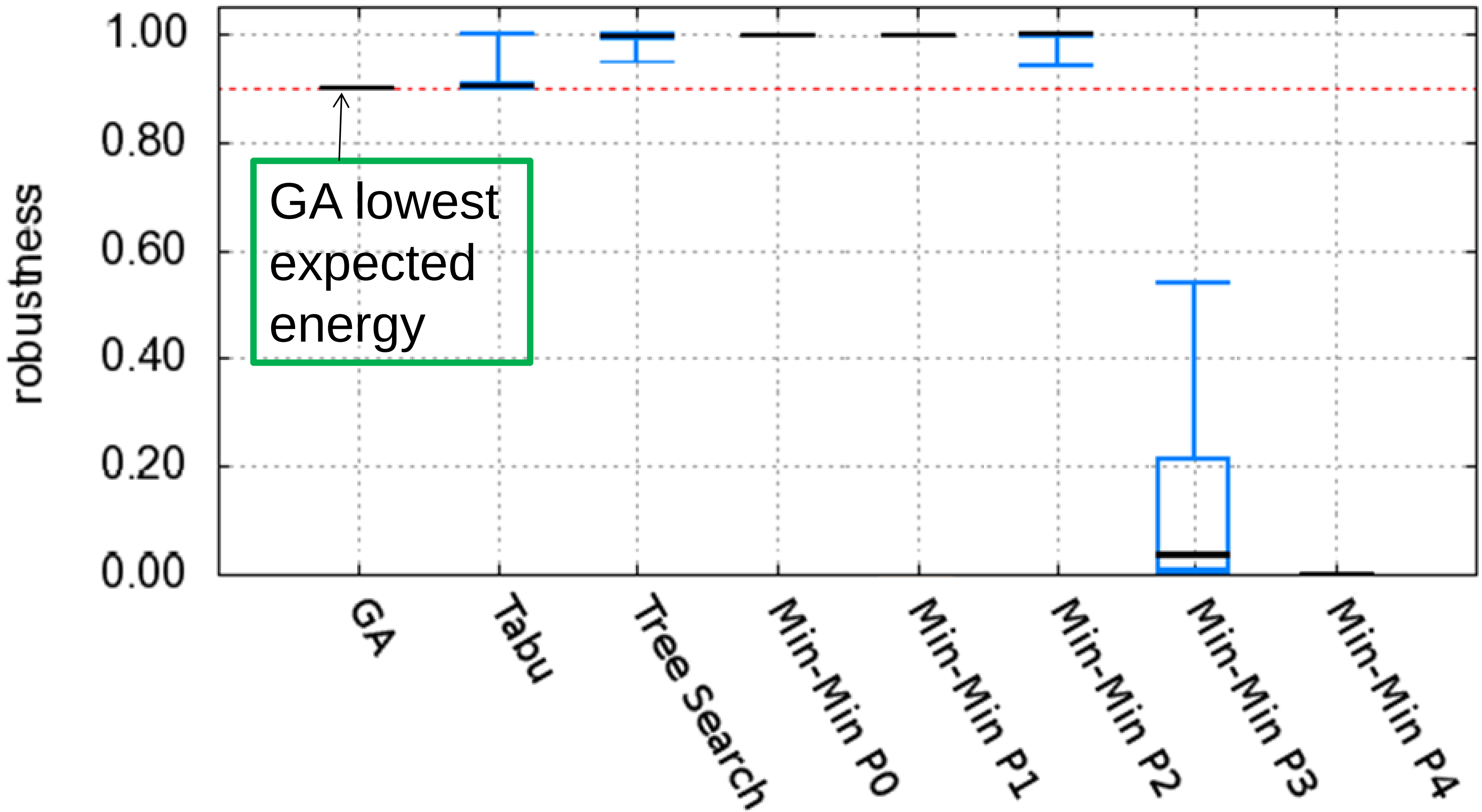
- 4000 tasks
- total of 25 compute nodes
- total of 63 multicore processors (randomly varied 1 to 4 per node)
- total of 178 cores (randomly varied 1 to 4 per processor)
- average overhead ~50% of total energy (varied across nodes)
- 90% probability constraint on finishing by deadline (robustness)
- GA parameters, determined by experimentation:
  - $p_c = 0.005$, $p_{tm} = 0.25$, $p_{tmg} = 0.001$, $p_{pm} = 0.025$, $p_{pmg} = 0.0005$
  - population size: 100
- 50 different simulation trials were run for each heuristic
  - different PMFs for task execution times

Results Static Study – Expected Energy



- box and whiskers: min, 1st quartile, median, 3rd quartile, max
- Min-Min in P-states 3 and 4 did not meet robustness constraint
- red: heuristic execution times

Results Static Study – Robustness



- robustness constraint (90%) shown as red dashed line
- no need to have robustness over 90%

Results Static Study – Discussion

- recall goal: design robust resource management techniques that
  - minimize expected energy used (performance measure)
  - constraint on probability of finishing by deadline (robustness)
- in general, lower performance P-states result in lower total expected energy (good) BUT lower robustness (bad)
  - use combination
- GA had lowest expected energy consumption and exactly met robustness constraint
- GA execution time per trial was 8 hours
  - not a problem because done off-line for static production environment

Outline

- stochastic model for resource allocation
- static resource allocation with energy minimization
- dynamic resource allocation with energy constraint
- conclusions

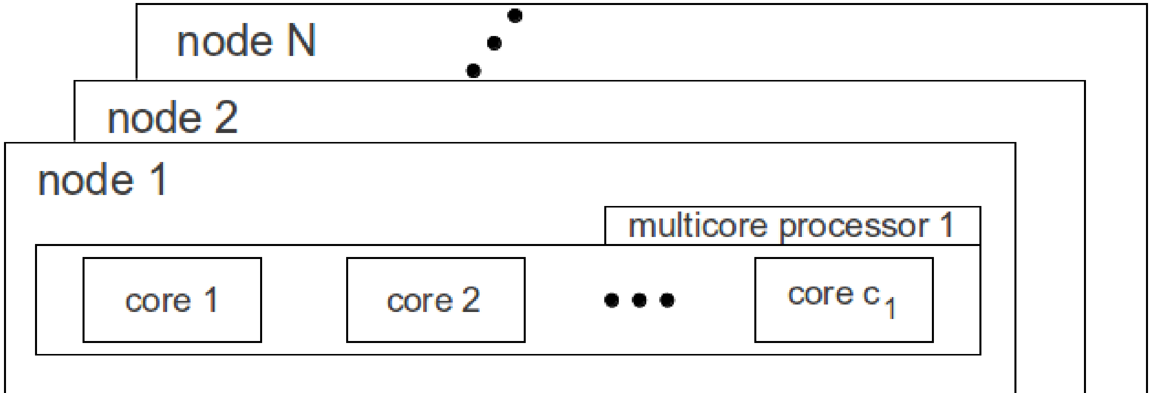
Colorado State University

Dynamic Heuristics with Energy Minimization

- reference
  - “Deadline and Energy Constrained Dynamic Resource Allocation in a Heterogeneous Computing Environment”
  - by Young, Apodaca, Briceño, Smith, Pasricha, Maciejewski, Siegel, Khemka, Bahirat, Ramirez, and Zou
  - *Journal of Supercomputing*
  - February 2013, Vol. 63, No. 2, pp. 326-347

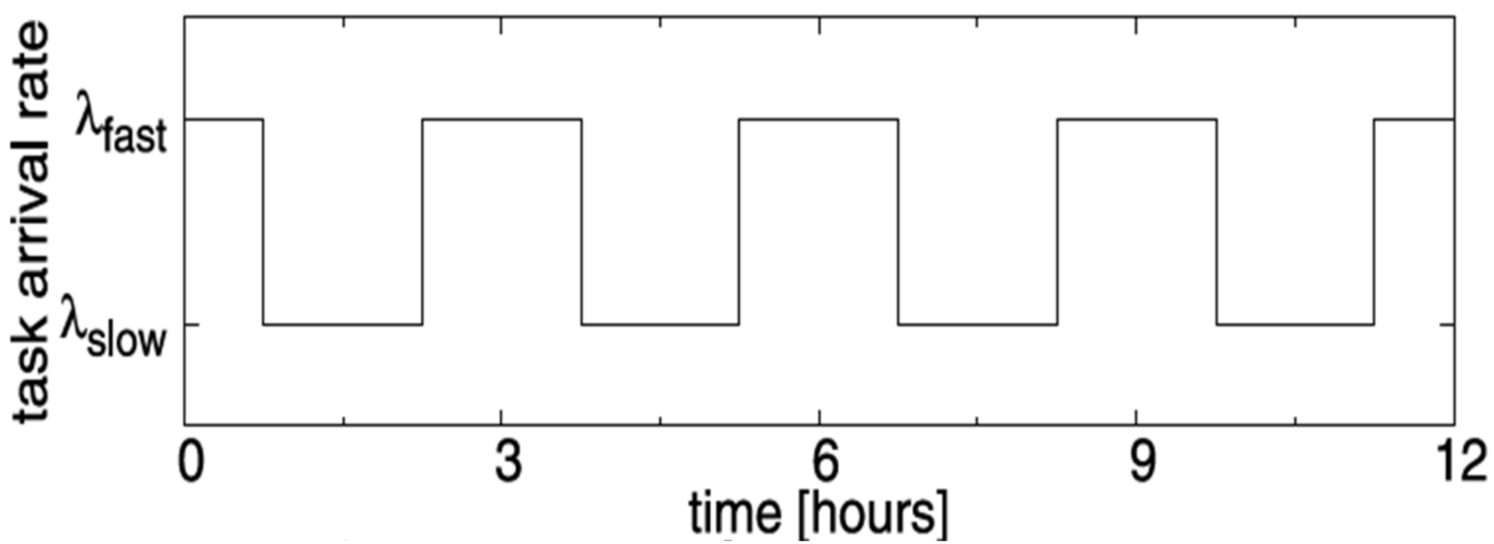
Problem Statement for Dynamic Study

- multi-core architecture similar to static study
  - cores always on
    - idle: P4
    - overhead power constant therefore not considered
- dynamic resource allocation
- goal: given a set of independent tasks with individual deadlines, design robust resource management techniques that
  - complete as many tasks as possible by their individual deadlines (performance measure)
  - subject to a constraint on total energy consumption



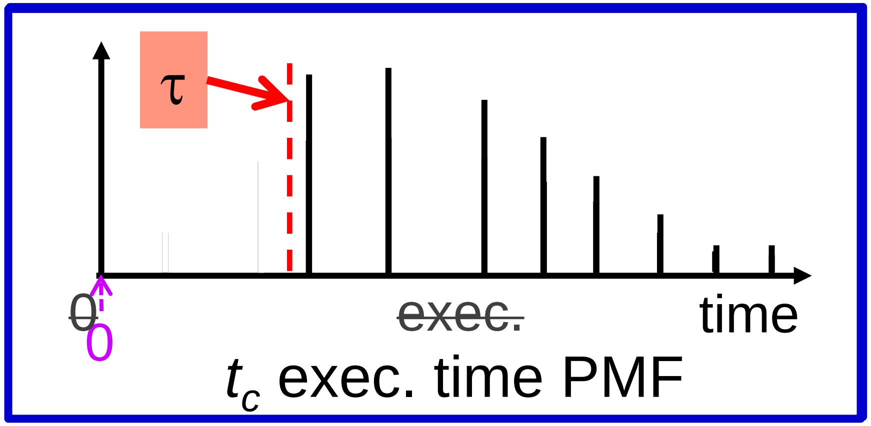
System Model for Dynamic Study

- dynamic, immediate-mode scheduler
  - each task scheduled when it arrives
- collection of known task types
- task type execution time per type of core represented by a PMF
  - can be found from historical data, experiments
- task arrivals modeled as two-phase Poisson process
  - oversubscribed: tasks arrive at a faster rate (**λ_{fast}**)
  - undersubscribed: tasks arrive at a slower rate (**λ_{slow}**)



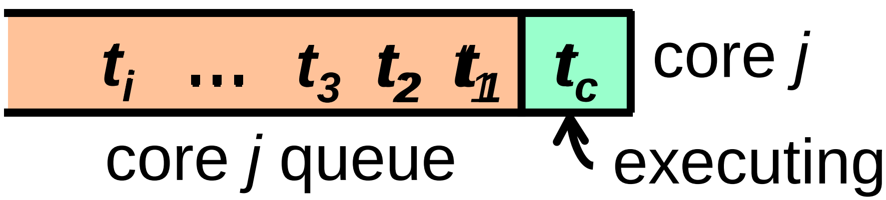
Completion Time PMF for Currently Executing Task

- assume currently executing task t_c is assigned to core j
- for task t_c
  - start with execution time PMF for that task on core j
  - shift PMF to begin at core j ready time
  - drop pulses less than current time τ
  - renormalize PMF



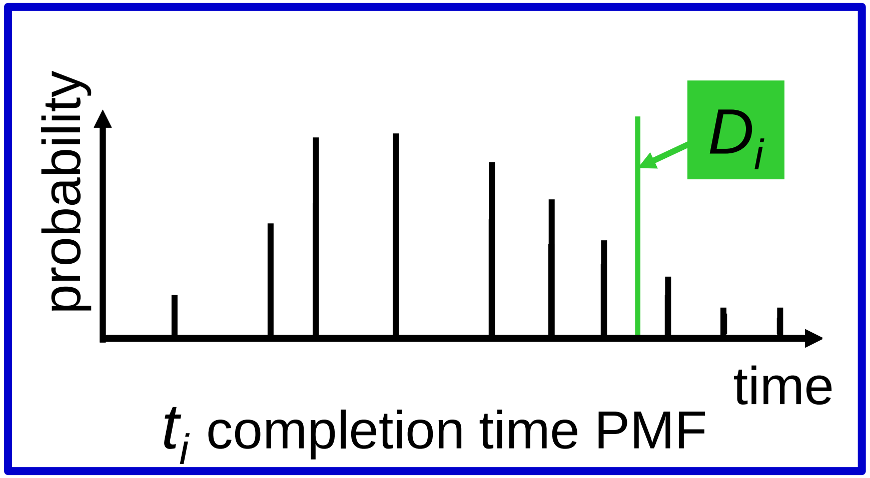
Completion Time PMF for a Task t_i

- to get task t_i completion time PMF ***convolve***
  - resulting completion time PMF for currently executing task t_c
  - execution time PMFs for all tasks queued ahead of task t_i on core j
  - execution time PMF for task t_i on core j



Expected Number of On-time Completions

- in resulting task t_i completion time PMF
  - sum pulses ≤ task t_i individual deadline D_i
  - this is probability task t_i will complete by its deadline



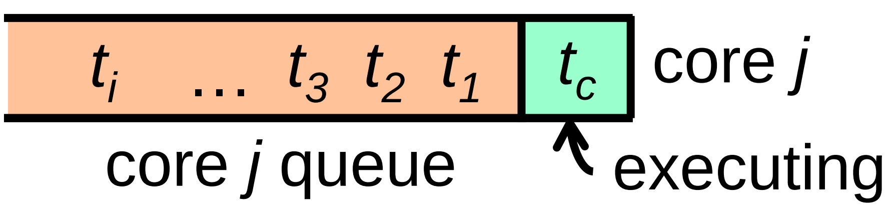
- sum "probability task will complete by its deadline" over all tasks – currently executing or queued
  - "expected number of on-time completions"
  - this is the measure of robustness

Robustness Definition for Dynamic Study

- **The Three Robustness Questions**
  1. what makes the system robust?
     - complete all tasks by their individual deadlines
  2. what uncertainties are the system robust against?
     - each task's execution times may vary substantially based on input data
  3. how is robustness quantified?
     - number of "expected on-time completions"

Start-Time Cancellation

- if start-time cancellation is used:
  - ready-to-execute task is cancelled if it has a probability of completing by its deadline below a **threshold**
    - tunable parameter by experimentation
    - 30% in this simulation study
  - to calculate t_i completion time PMF
    - omit any task queued ahead of task t_i whose probability of meeting its deadline < threshold
    - just prediction that it will be cancelled



- a task cannot be stopped once execution started

Heuristics for Dynamic Study

- recall goal: set of independent tasks with <u>individual</u> deadlines
  - complete as many tasks as possible by their <u>individual</u> deadlines (performance measure)
  - subject to a constraint on total energy consumption
- assign each task to a node, multi-core processor, core, and P-state when it arrives (immediate mode)
- can use filters to add energy and/or robustness awareness
- may leave tasks unassigned or cancel a task
- heuristics from the paper
  - Lightest Load
  - Minimum Expected Completion Time
  - Shortest Queue
  - Random (for comparison)

Heuristic: Lightest Load

- attempt to balance energy and robustness by minimizing a "load" *L*
- for a given task, consider its *L* value for each core and P-state
- En_{ex}: expected energy consumed for the assignment
  - product of the expected execution time and the power consumption
- *p*: probability of the task completing by its deadline for the assignment
- $L = (100 - p) \times En_{ex}$
  - smaller is better
  - when $p = 100$ then En_{ex} is effectively ignored
    - frequently occurs during end of undersubscribed periods
- assign incoming task to the core/P-state combination with the smallest *L* value

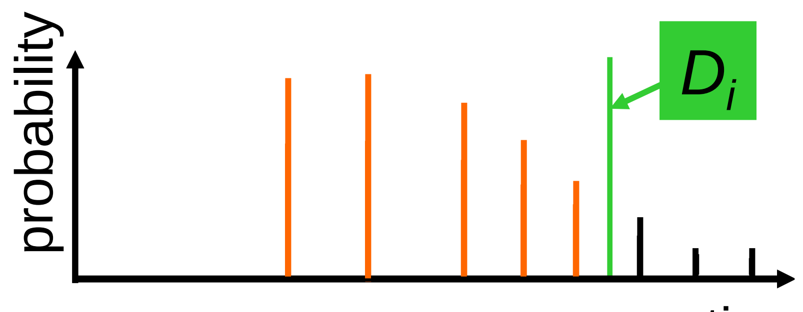
Energy Filter Based on Estimated Remaining

- restrict potential core/P-state assignments to those $\leq$ energy threshold $En_{thresh}(t)$ at time step t
  - discard task if no assignment meets threshold
- $En_{mul}(t)$:
  - three fixed values: for different levels of average queue depth
  - found empirically using a subset of simulation trials
- $t_{rem}(t)$: time remaining in the 12−hour simulation trial
- $En_{rem}(t)$: estimated energy remaining in 12-hour interval
  - energy constraint minus (expected for queued plus “simulated actual” for completed)
- c_{tot}: total number of cores in the system
- t_{avg}: average task execution time

- $$En_{thresh}(t) = En_{mul}(t) \times \overbrace{En_{rem}(\mathrm{t})/\underbrace{[\overbrace{(t_{rem}(t)/t_{avg})}^{\text{\# tasks/core}} \times c_{tot}]}_{\text{total \# tasks}}}^{\text{fair share per task}}$$

Robustness Filter

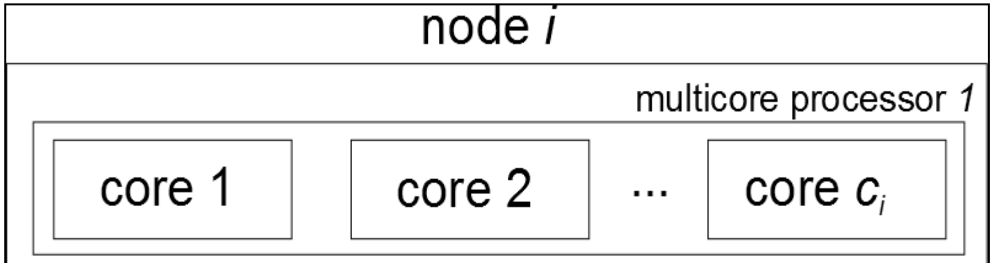
- based on the task's contribution to the total robustness measure
- restrict potential assignments using a robustness threshold p_{thresh} on the probability of the task completing by its individual deadline
  - limits assignments to those that will increase the expected number of on-time completions (robustness) by at least the threshold
  - threshold found empirically (simulation study: $p_{\text{thresh}} = 30\%$)
  - discard task if no assignment meets threshold



t_i completion time PMF

Simulation Setup for Dynamic Study

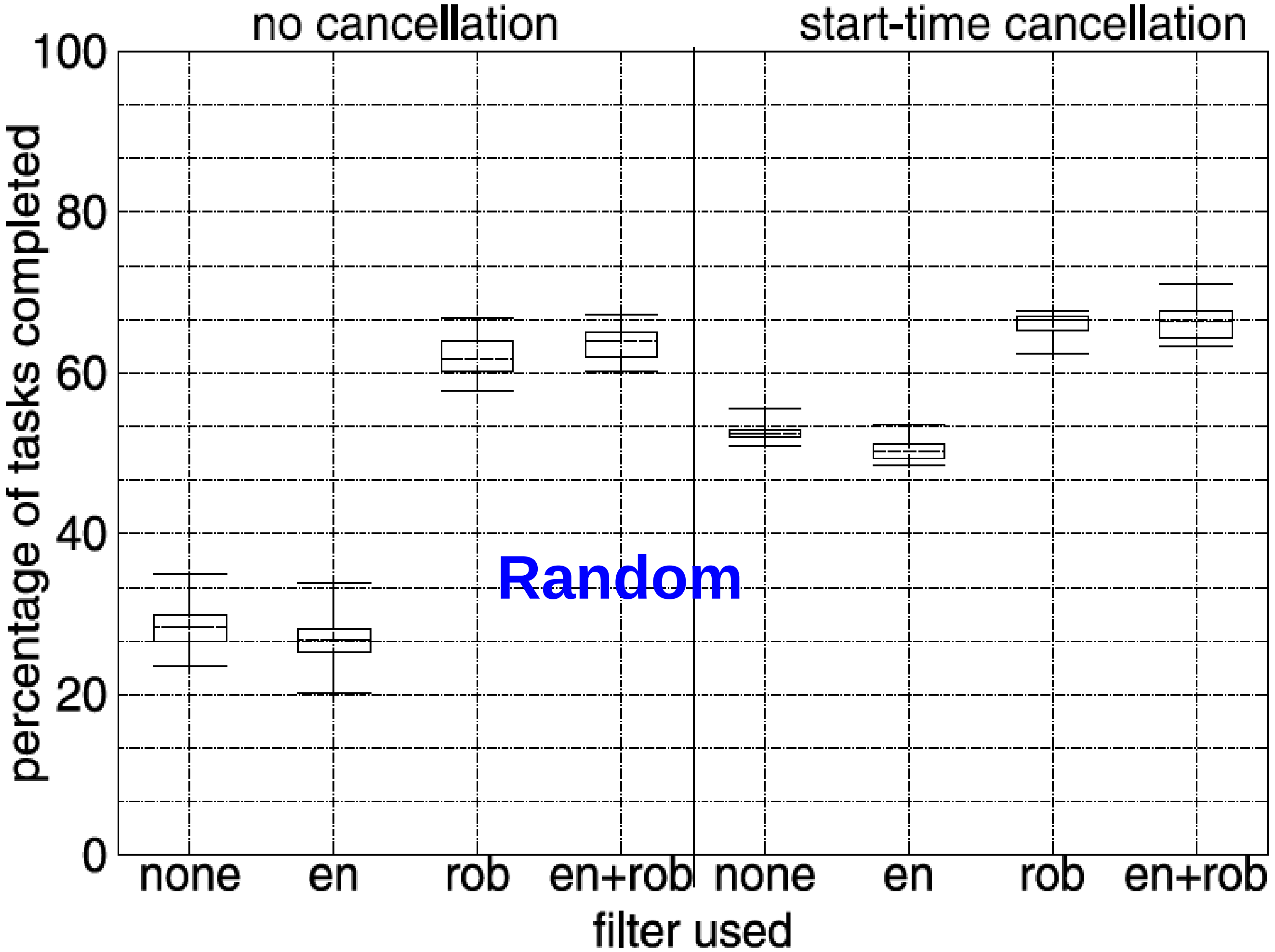
- 12-hour trial window, ~1,650 tasks, 100 task types
- total of 25 compute nodes
- total of 63 multicore processors (varied 1 to 4 per node)
- total of 178 cores (varied 1 to 4 per processor)



- variations among 50 simulation trials:
  - task-type mix
  - task arrival times
  - task "simulated actual" execution times (sample PMFs)
- individual deadline = arrival time + average execution time of its task type over all machines & P-states + average over all tasks
  - tight deadlines

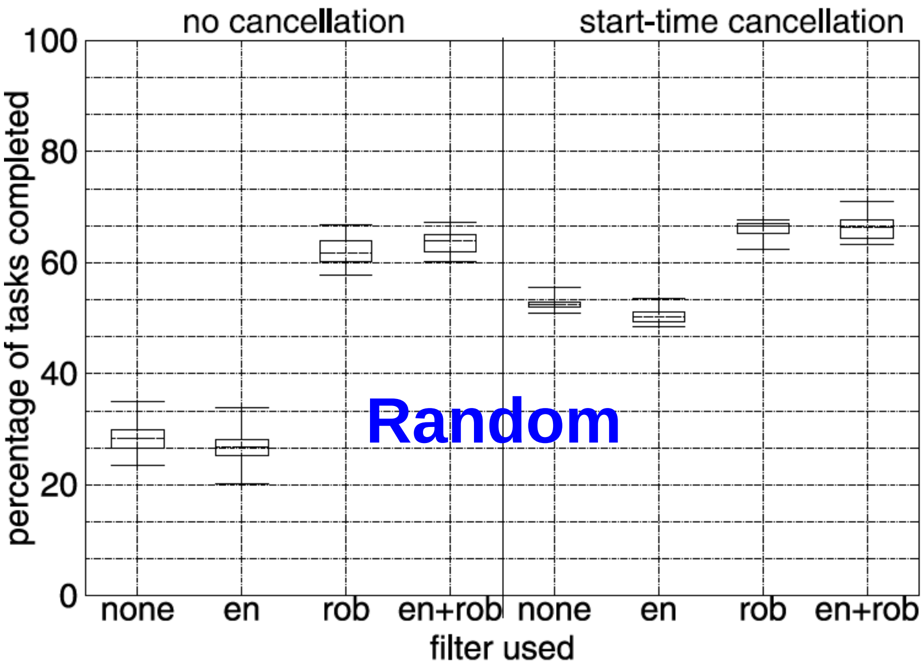
Results: Impact of Filters and Cancellation

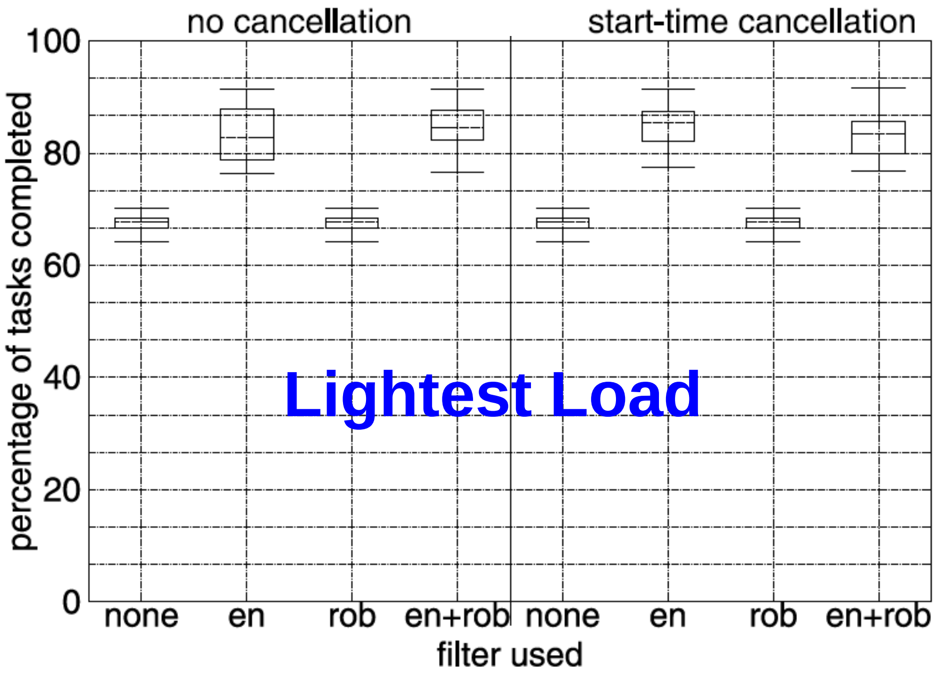
- randomly assign incoming task to a random core and P-state
- “robustness filter” and “energy + robustness filter” and “start time cancellation” improve over random assignment



Results: Random versus Heuristic

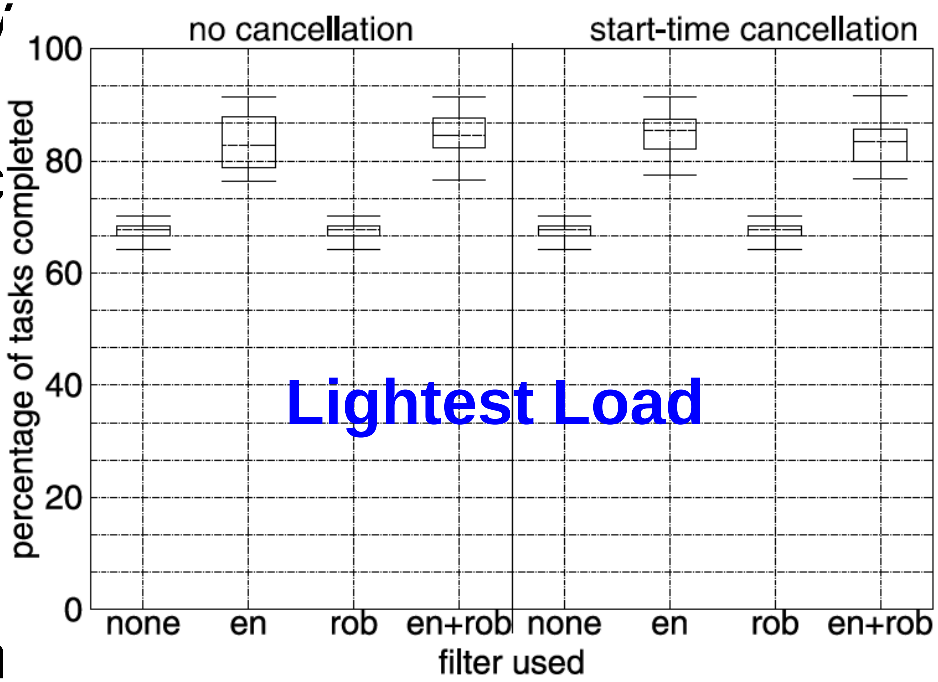
- heuristics better than Random





Results: Dynamic Study Discussion

- heuristics performed comparably
- filters together better than none
- filters more impact than heuristic choice or cancellation
- "robustness filter" little impact
  - probability task meeting its individual deadline
  - eliminating mappings that would not have been chosen



- "energy filter" ensures energy left for tasks that arrive later
- start-time cancellation has limited impact
  - heuristic already considers task execution times
  - difficult for start-time cancellation to predict perfectly

Outline

- stochastic model for resource allocation
- static resource allocation with energy minimization
- dynamic resource allocation with energy constraint
- conclusions

Current and Future Research

- explore **stochastic robustness** allocation **heuristics** for different
  - static and dynamic
  - performance measure, constraints
  - workload and platform characteristics
- use **energy** or **power** as performance metric or constraint
- consider that **cost of power** may **vary** during day
- impact of DVFS for **memory vs. compute intensive** tasks
- study interaction of energy and **time-dependent utility** functions
- combine **multiple** uncertainties in single robustness measure
- combining PMFs/probabilities when **not independent** (ex**. DAG**)
- how to be robust with respect to **inaccuracies** in the PMFs
- model conflicts due to **resource sharing** in multi-core systems
- **thermal-aware** resource management
- **multi-objective optimization** of energy/power and QoS

Colorado State University

Concluding Remarks

- **The Three Robustness Questions**
  1. what behavior of the system makes it robust?
  2. what uncertainties is the system robust against?
  3. how is robustness of the system quantified?
- presented a stochastic model for robust resource allocation
- used stochastic robustness in energy-aware resource allocation
- listed areas for future research
- for more information and references to other relevant research
  www.engr.colostate.edu/~hj/Robust_Papers.pdf