Robust Computing Systems

Resource Management for Heterogeneous Computing Systems

H. J. Siegel, Professor Emeritus
Colorado State University
Formerly:
Abell Endowed Chair Distinguished Professor
of Electrical and Computer Engineering
and Professor of Computer Science

Dilbert Feb 14, 2010

Robust Computing Systems

Resource Management for Heterogeneous Computing Systems

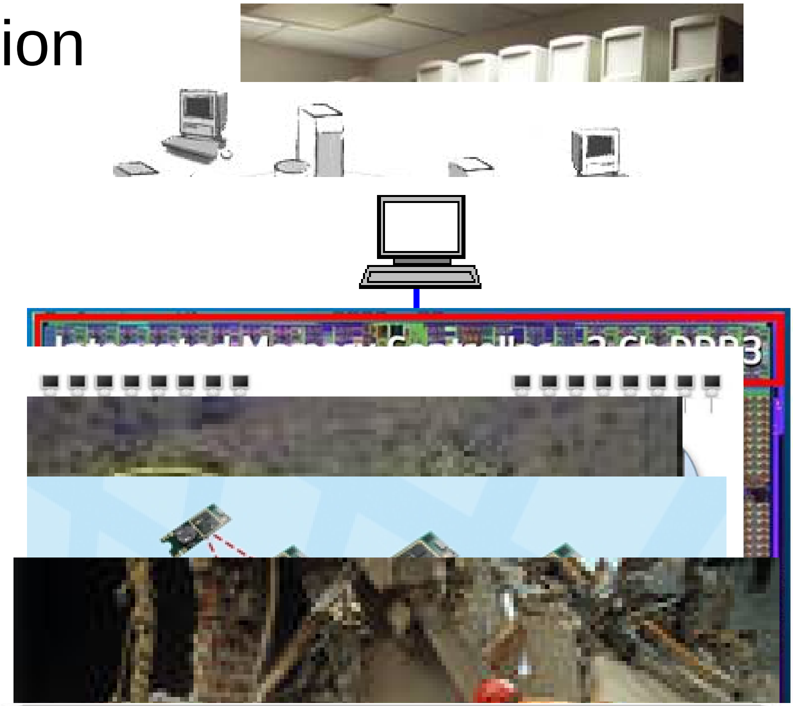
H. J. Siegel, Professor Emeritus
Colorado State University
Formerly:
Abell Endowed Chair Distinguished Professor
of Electrical and Computer Engineering
and Professor of Computer Science

Outline

- definition and stochastic model of robustness
- use in static resource allocation heuristics
- use in dynamic resource allocation heuristics
- summary and concluding remarks

Applicability of Stochastic Robustness Model

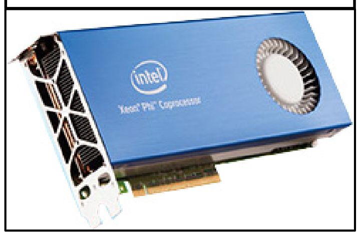
- variety of computing and communication environments, such as
  - cluster
  - grid
  - cloud
  - content distribution networks
  - wireless networks
  - sensor networks
- design problems throughout various scientific and engineering fields
  - examples we are exploring
    - search and rescue
    - smart grids



Heterogeneous Computing System

- interconnected machines with different computational capabilities
- workload of tasks with different computational requirements
  - heterogeneity to service diverse computational workloads
- each task may perform differently on each machine
  - machine A better than machine B for task 1 but not for task 2
- research also applies to a cluster of different types
  (or different ages) of machines, grids, and clouds

Intel Phi Coprocessor

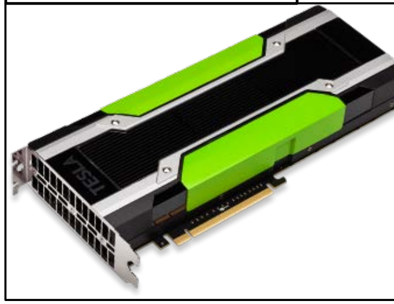


Hitachi Blade Server 500

Cray XC-30 Blades

HP BladeSystem C7000

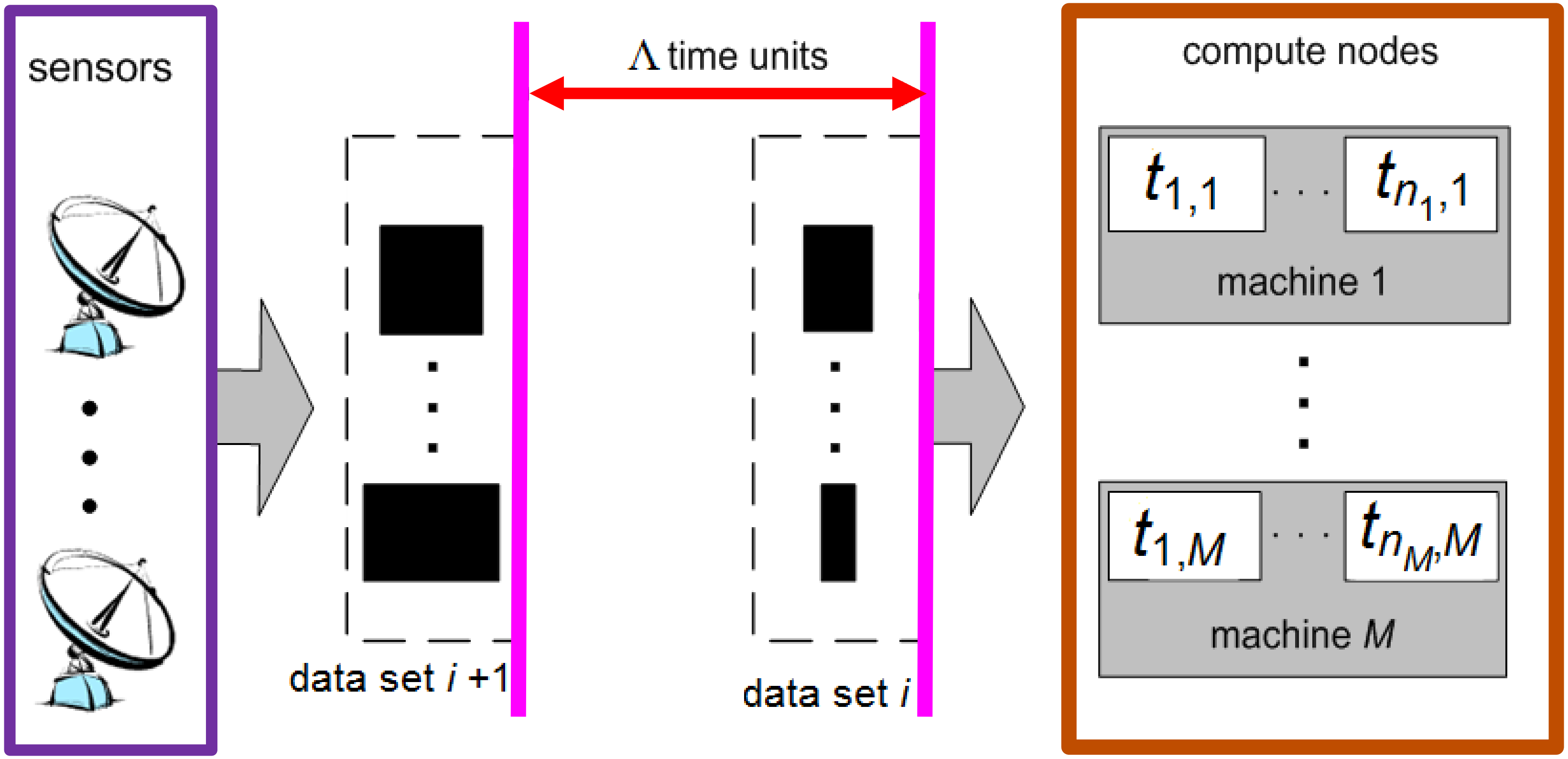
Nvidia Tesla GPU



Resource Management

- assign and schedule (map) tasks to machines
  - optimize some performance measure
  - possibly under a system constraint
- in general, known **NP-Hard** 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 > 20,000 years!
  - use **heuristics** to find near-optimal solutions

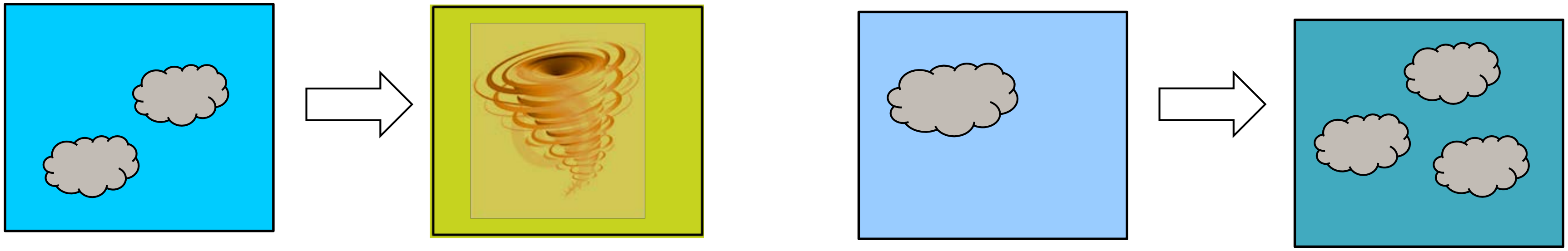
Ex.: Radar Data Processing for Weather Forecasting



- sensors produce periodic data sets, each with multiple data files
- ***N*** independent tasks process each data set within **Λ** time units
- *N* tasks **statically** mapped to ***M*** heterogeneous machines, $N > M$
- similar: satellite data maps, security surveillance

Uncertainty in Environment

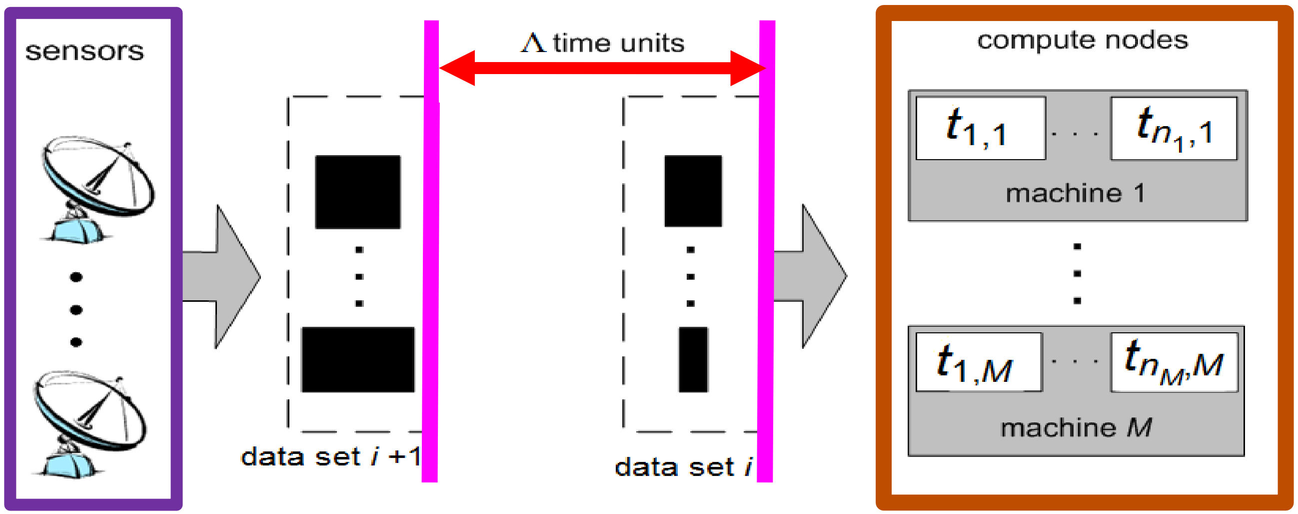
- **variability** across the data sets results in variability of the execution time of each task even on the same machine
  - examples
    - types of objects found in a radar scan data file
    - increase in number of objects in a radar scan data file



- unable to predict exact execution times of tasks
  - **uncertainty** parameters in the system
  - **history** of task exec times on each machine, different data
- use history to find allocation that is **robust** against uncertainty

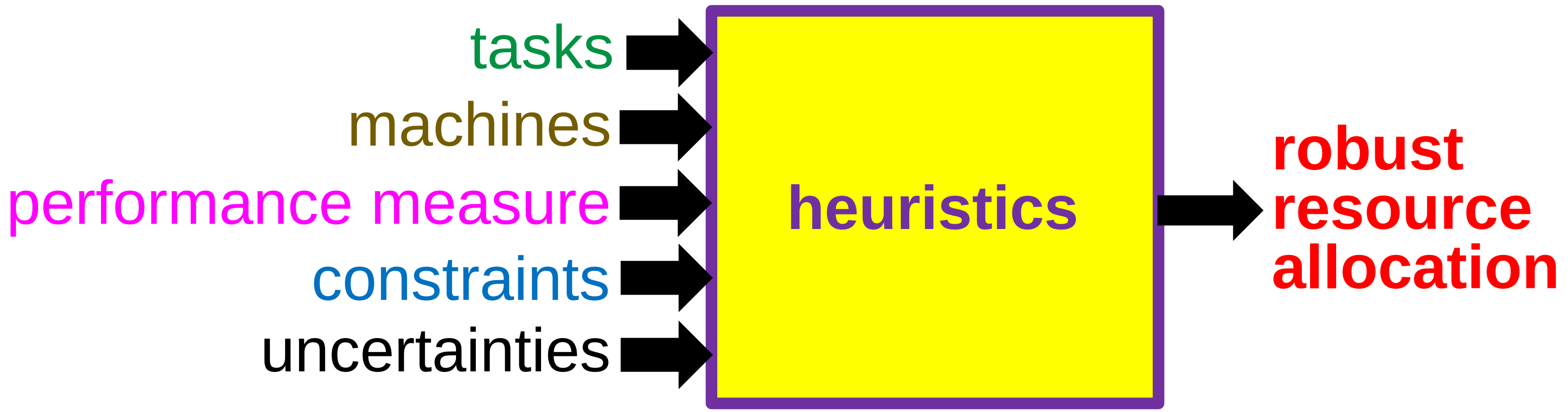
Problem Statement for Static Resource Allocation

- unpredictable execution times of the tasks across data sets
- have a probabilistic guarantee of performance of a mapping
- **problem statement**
- determine a **robust** static resource allocation
  - goal: minimize time period (Λ) between data sets
  - constraint: a user-specified probability of 90% that all tasks will complete in Λ time units for each data set



Problem Statement for Static Resource Allocation

- unpredictable execution times of the tasks across data sets
- have a probabilistic guarantee of performance of a mapping
- **problem statement**
- determine a **robust** static resource allocation
  - goal: minimize time period (Λ) between data sets
  - constraint: a user-specified probability of 90% that all tasks will complete in Λ time units for each data set



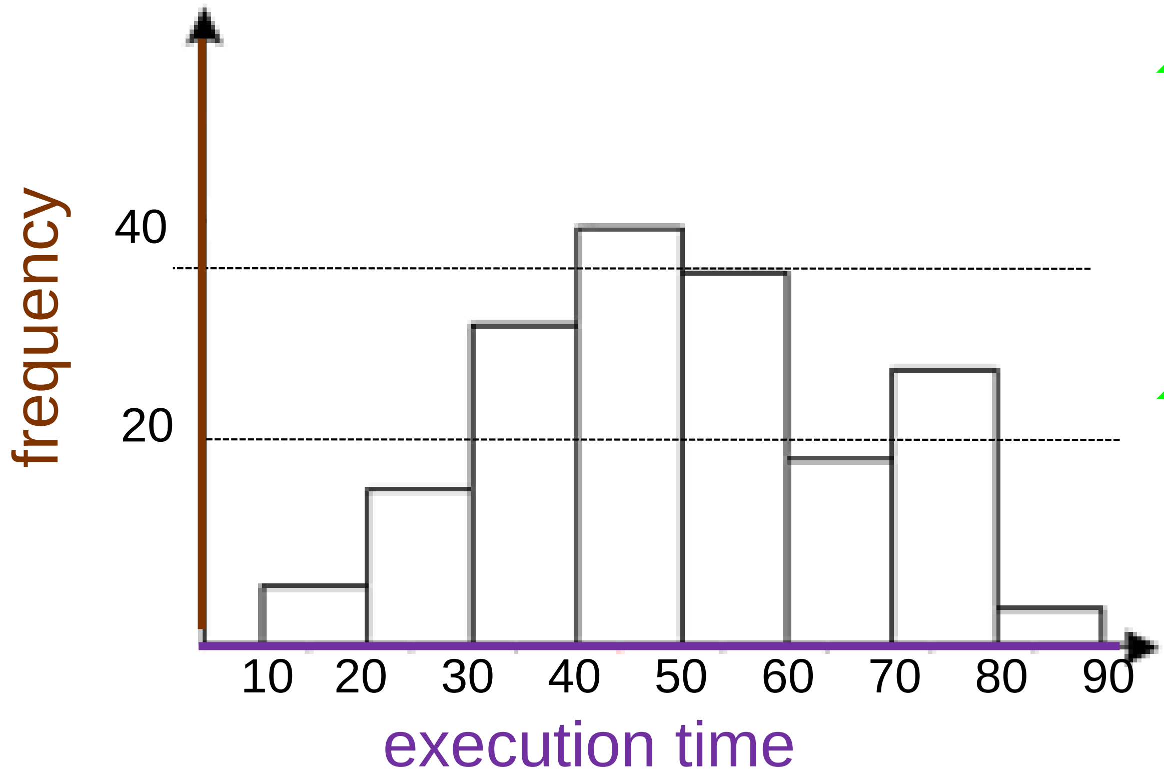
Defining Robustness for Static Resource Allocation

- term "robustness" usually used without explicit definition
- three general robustness questions that should be answered
- **THE THREE ROBUSTNESS QUESTIONS**
  1. what behavior of the system makes it robust?
     - ex. execute all tasks within Λ time units
  2. what uncertainty is the system robust against?
     - ex. execution times of tasks vary over different data sets
  3. how is robustness of the system quantified?
     - ex. probability that the resource allocation will execute all tasks within Λ time units for every data set

Colorado State University

Modeling Uncertain Task Execution Times

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

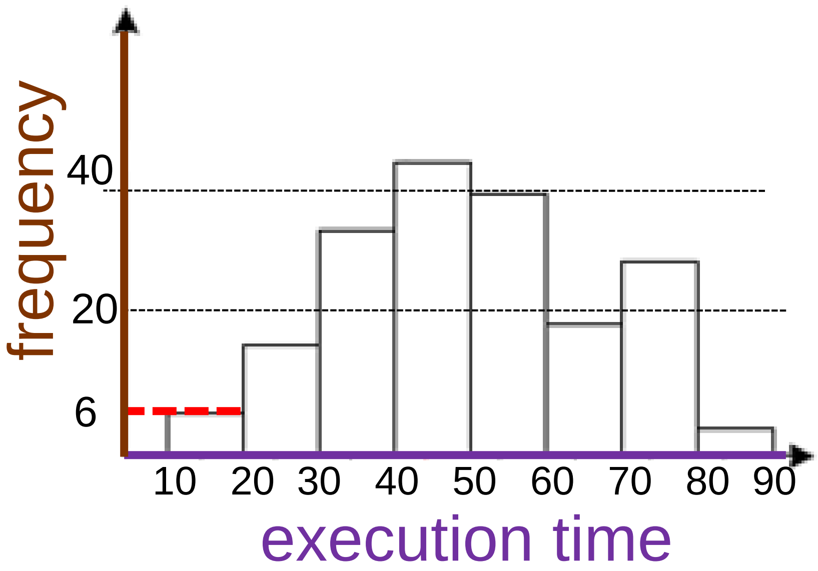


- x-axis: execution time within 10 second interval bins
- y-axis: frequency = height of bar for a given interval

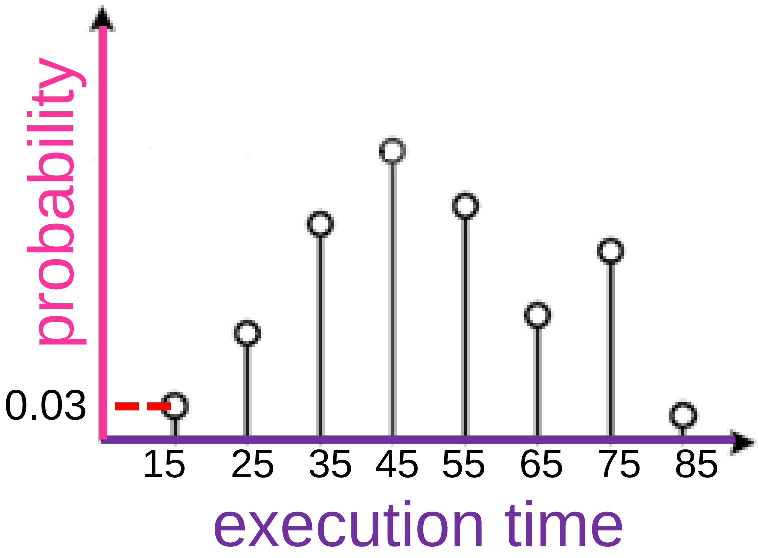
Generating a PMF from a Histogram

- generate probability mass function (PMF) 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%

histogram

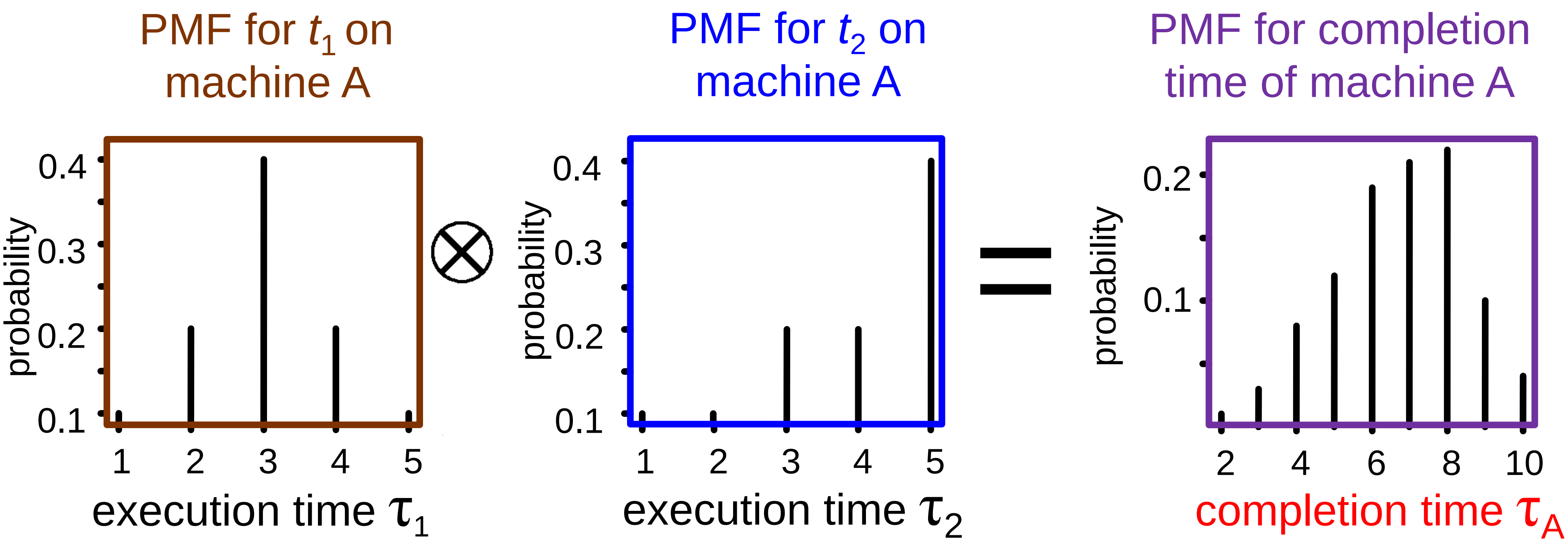


PMF



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
- “convolution” of the execution time PMFs for two tasks

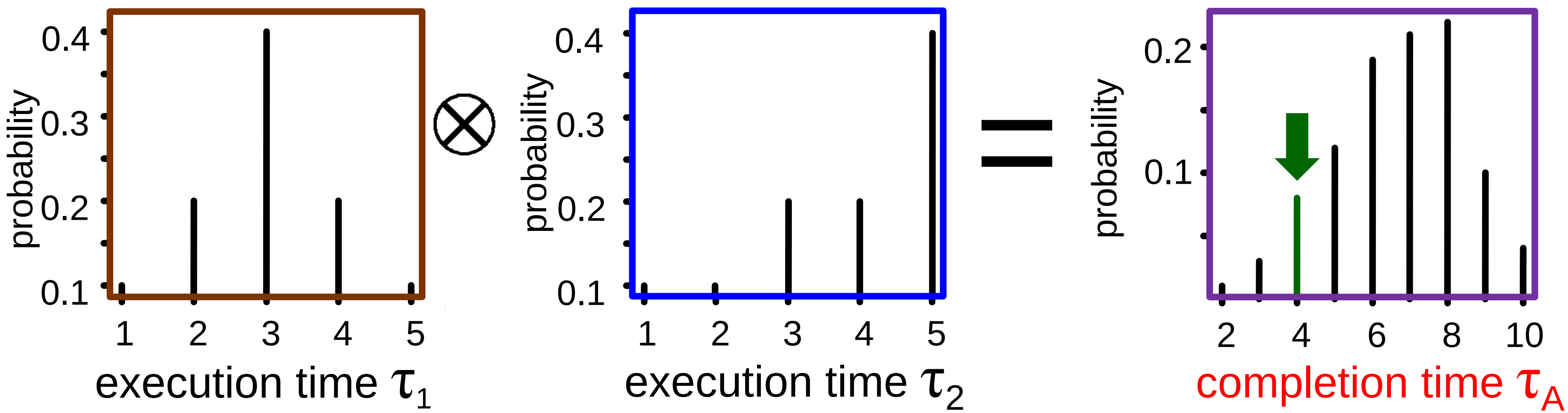


- $$p(\tau_A = k) = \sum_{\tau_1+\tau_2=k} \left(p(\tau_1)\cdot p(\tau_2)\right)$$

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
- "convolution" of the execution time PMFs for two tasks

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



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

4

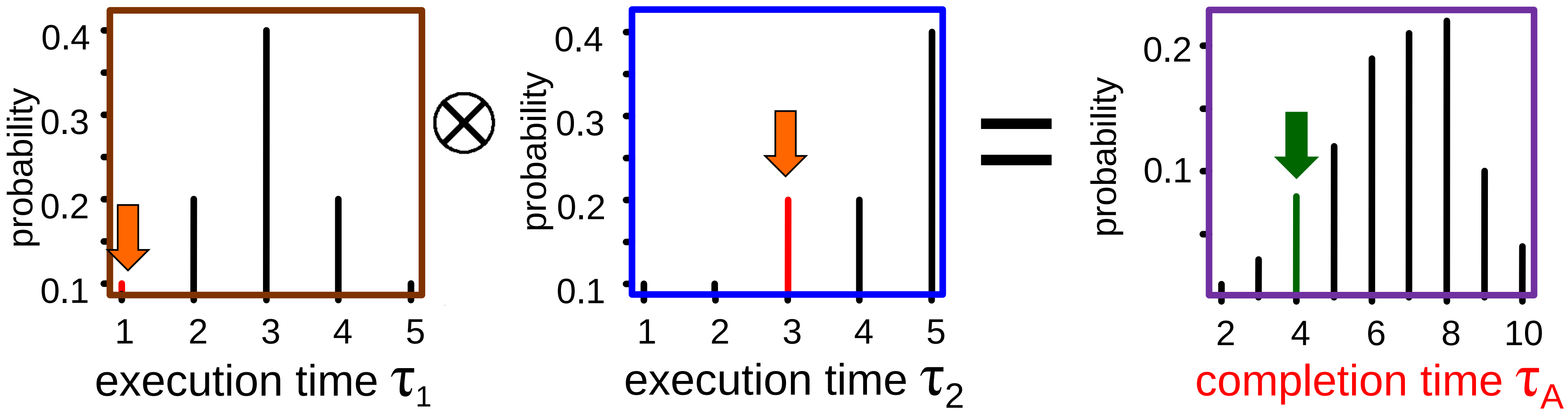
PMF for Completion Time of Machine

- assume task 1 and task 2 only tasks assigned to machine A
- can find completion time PMF for machine A to do both tasks
- "convolution" of the execution time PMFs for two tasks

PMF for t_1 on machine A

PMF for t_2 on machine A

PMF for completion time of machine A



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

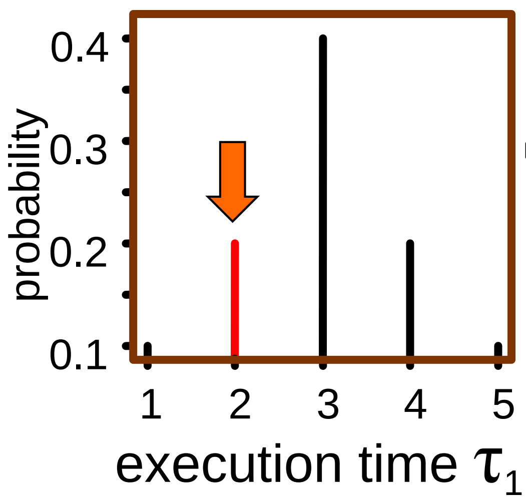
4

1 + 3 = 4

PMF for Completion Time of Machine

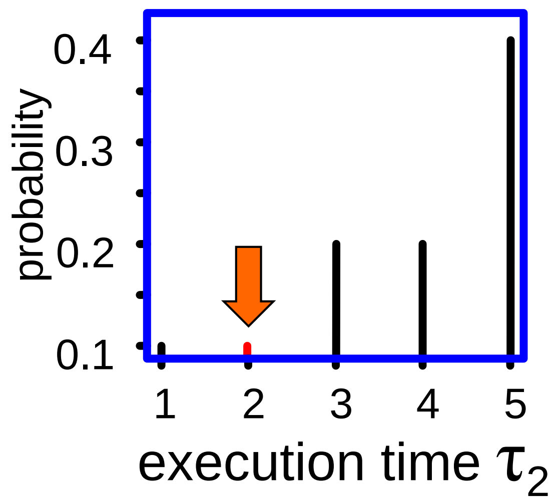
- assume task 1 and task 2 only tasks assigned to machine A
- can find completion time PMF for machine A to do both tasks
- "convolution" of the execution time PMFs for two tasks

PMF for t_1 on machine A



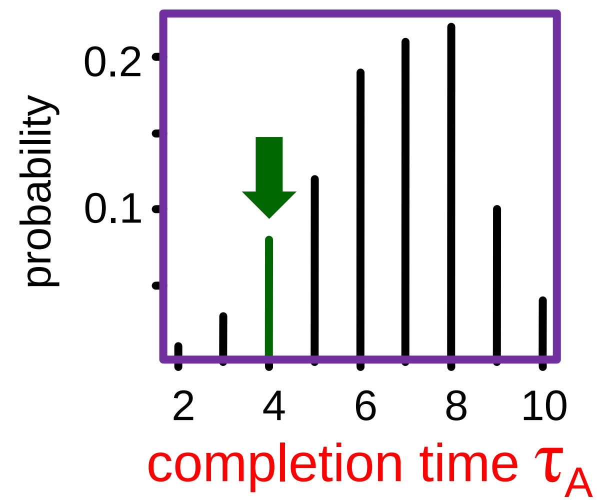
$\otimes$

PMF for t_2 on machine A



=

PMF for completion time of machine A



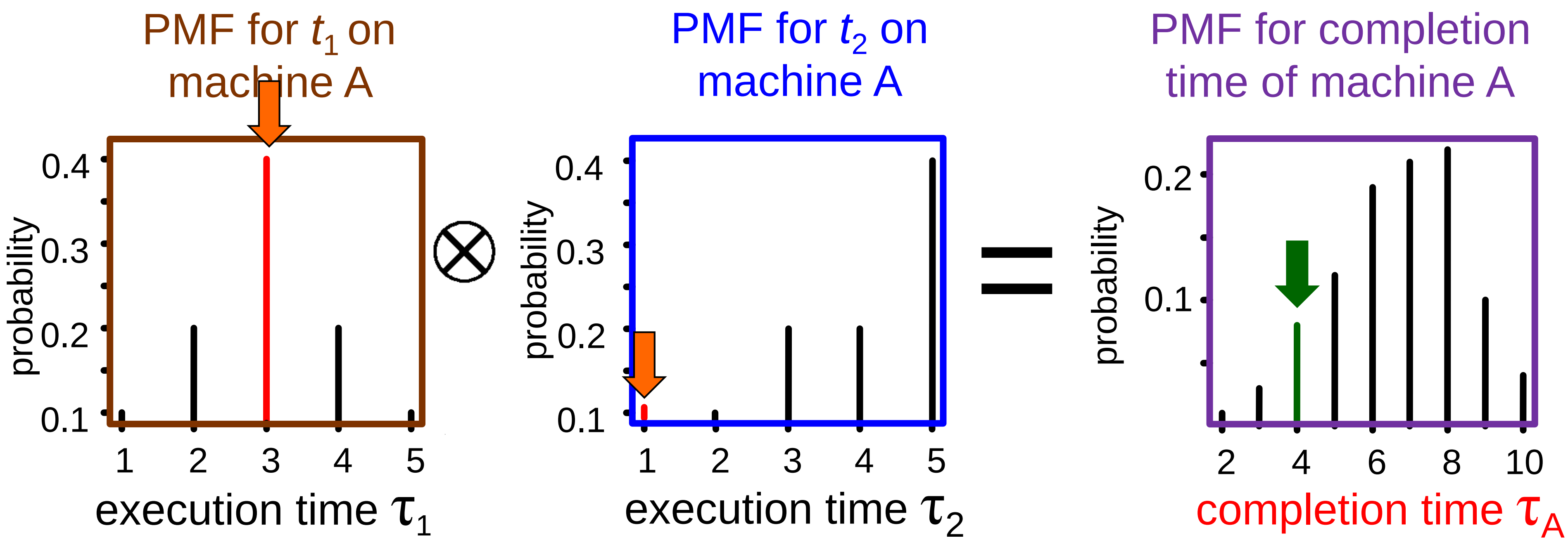
- $$p(\tau_A = k) = \sum_{\tau_1+\tau_2=k} (p(\tau_1)\cdot p(\tau_2))$$

4

2 + 2 = 4

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
- "convolution" of the execution time PMFs for two tasks



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

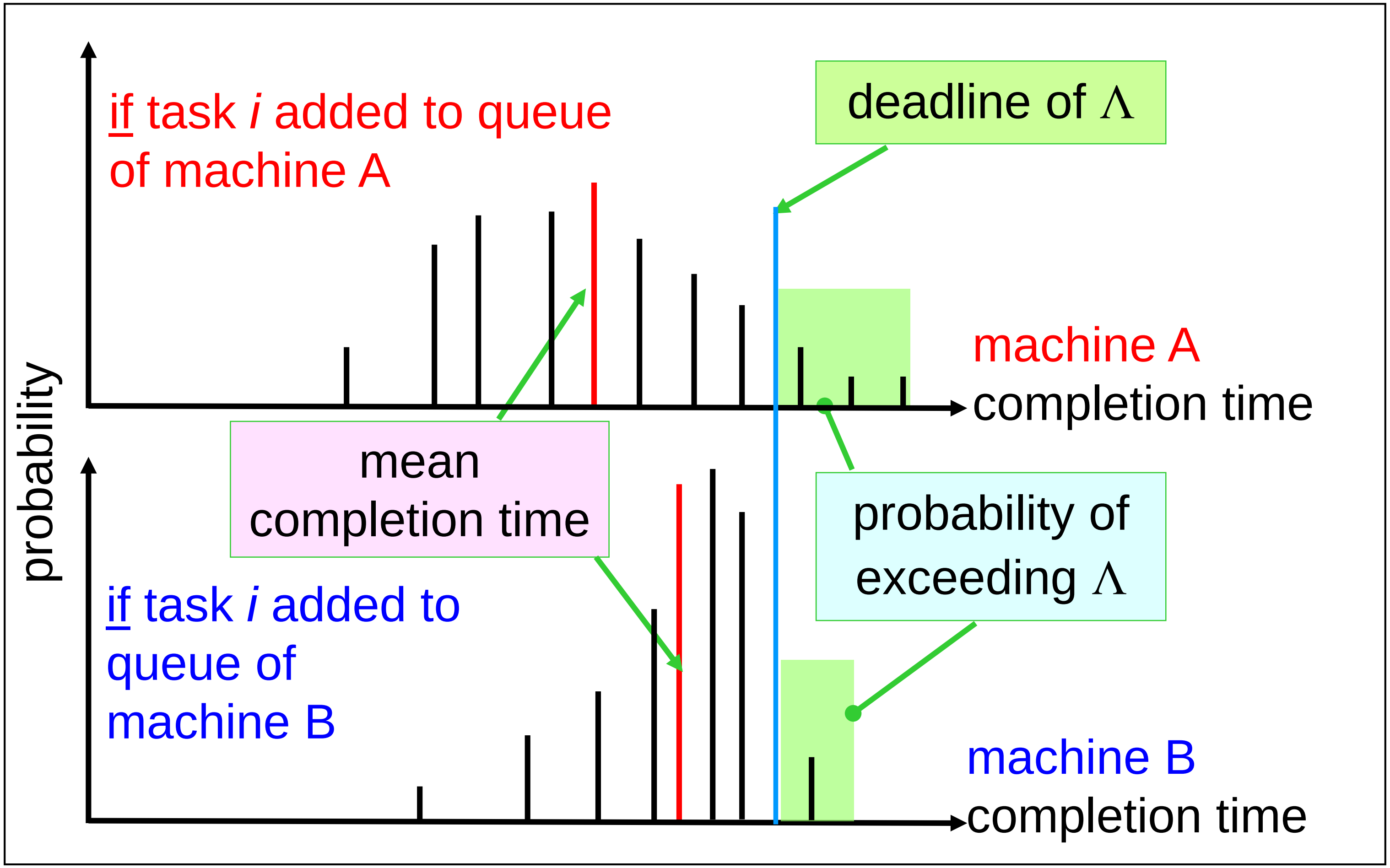
4

3 + 1 = 4

Colorado State University

Example of Use of Stochastic Model in Allocation

- PMFs for machine completion time based on
  - PMFs for tasks already assigned to that machine
  - 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

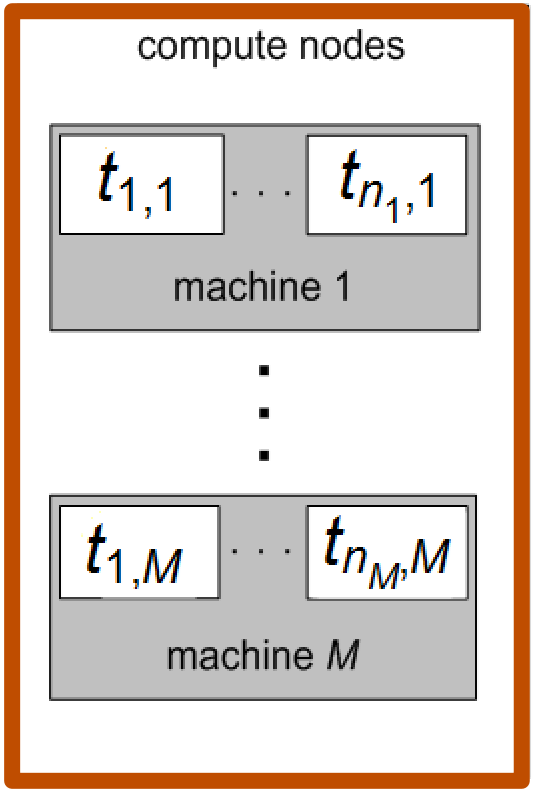
Colorado State University

Stochastic Robustness Heuristic Goals

- Λ : deadline for completing all tasks
- machine *j* stochastic robustness Prob[$S_j \leq \Lambda$]
- **Stochastic Robustness Metric (SRM)**

$$\prod_{j=1}^{M} \text{Prob}[S_j \leq \Lambda]$$

- **goal of heuristics**
  - minimize <u>Λ</u> for a given <u>SRM</u> value



Outline

- definition and stochastic model of robustness
- **use in static resource allocation heuristics**
- use in dynamic resource allocation heuristics
- summary and concluding remarks

Heuristic: Two-Phase Greedy Heuristic

- **problem:** static assignment of *N* tasks to *M* machines
  - minimize Λ for a given SRM value, for example 90%
- **while** there are still mappable tasks
  - **phase 1:** for each of the mappable tasks
    - find machine assignment for minimum Λ
  - **phase 2:** among these task/machine pairs
    - find task/machine pair with minimum Λ
    - map this task to its associated machine

Colorado State University

Heuristic: Genitor Genetic Algorithm

- chromosome of length *N* (number of tasks) = a mapping (solution)
  - *i* th element identifies the **machine** assigned to **task *i***

1	2	3	4	5	6	7	8	9	10	. . .
2	1	2	3	1	2	3	1	2	2	. . .

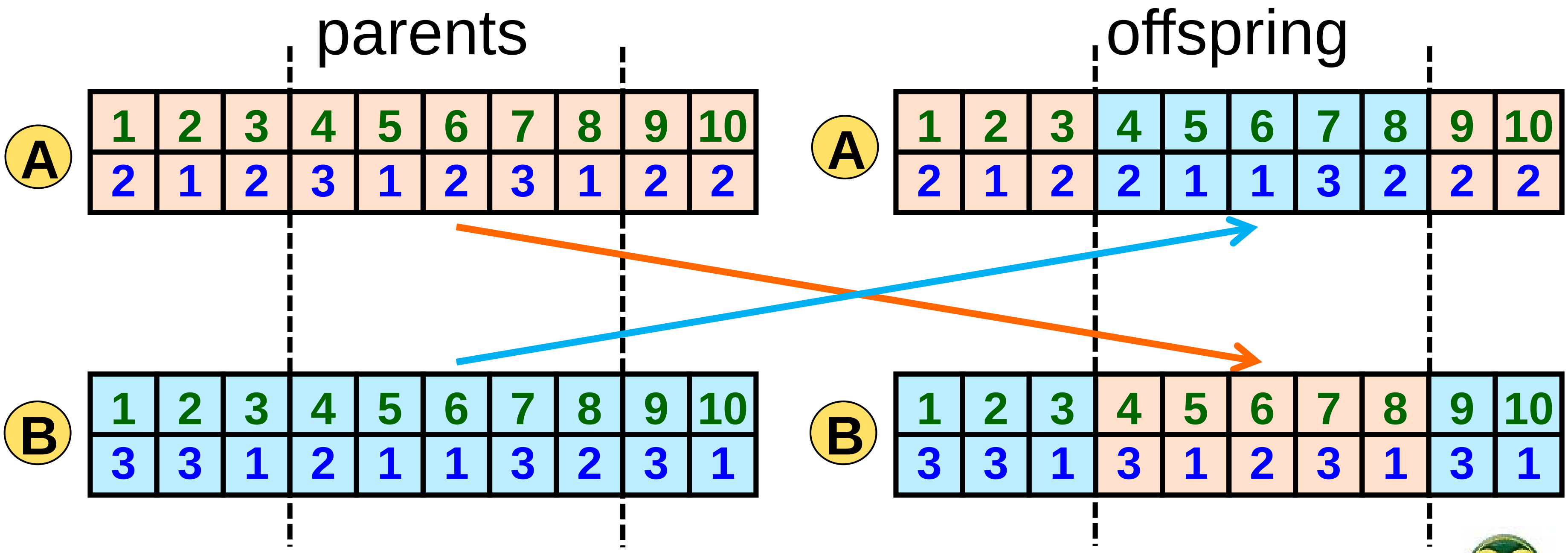
- population size of 200 (decided empirically)
- initial population generation
  - one chromosome: solution from the Two-Phase Greedy heuristic ("seed")
  - other 199: simple greedy heuristic
- population in ascending order based on minimum Λ value for given SRM (probability)

Procedure for Genitor

- **while** stopping criterion
  - select two parent chromosomes from population
  - perform crossover
  - **for** each offspring chromosome
    - perform mutation
    - apply local search
  - insert offspring into population based on minimum Λ order
  - trim population to population size
- **end of while**
- output the best solution

Genitor: Crossover

- selection of parents is done probabilistically
- crossover points are randomly selected
- exchange elements between crossover points
- generates two offspring



Genitor: Mutation

1	2	3	4	5	6	7	8	9	10	...
2	1	2	3	1	2	3	1	2	2	...

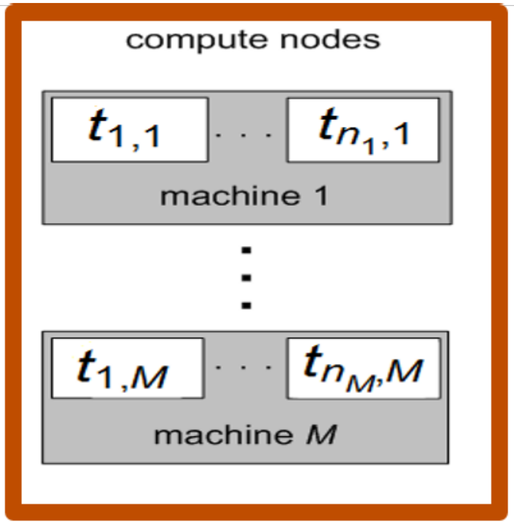
5

- mutation applied to offspring obtained from the crossover
  - for each element of each offspring chromosome
    - assignment has a 1% probability of mutation
  - mutation randomly selects a different machine

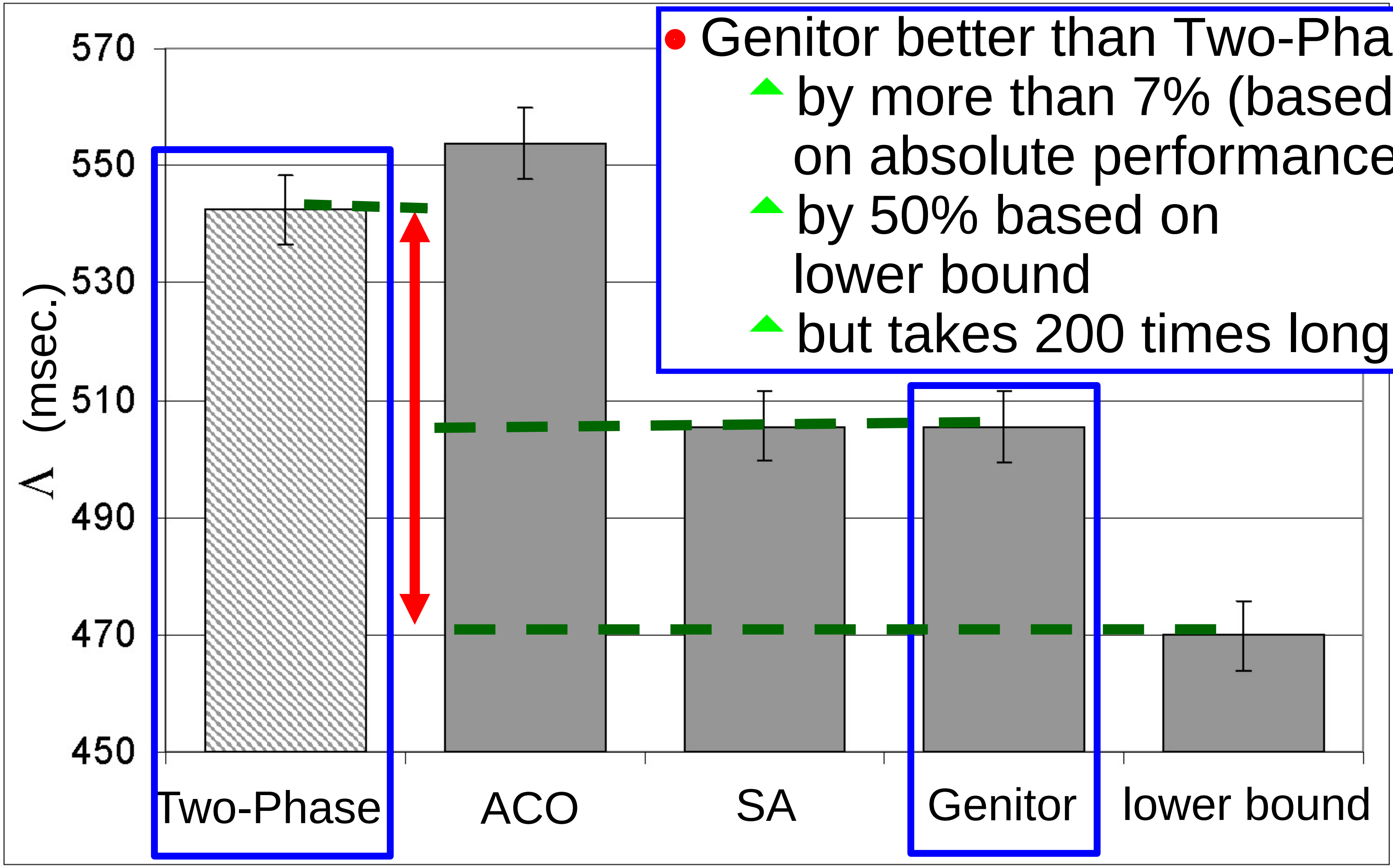
Colorado State University

Genitor: Local Search

- local search applied to each offspring
  - 1. for machine with individual highest Λ
    - consider moving each task to other machines
    - if improvement, move the task that gives smallest overall system Λ
  - 2. repeat 1 until no more improvement



Simulations: Performance of Static Heuristics



- Genitor better than Two-Phase
  - by more than 7% (based on absolute performance)
  - by 50% based on lower bound
  - but takes 200 times longer

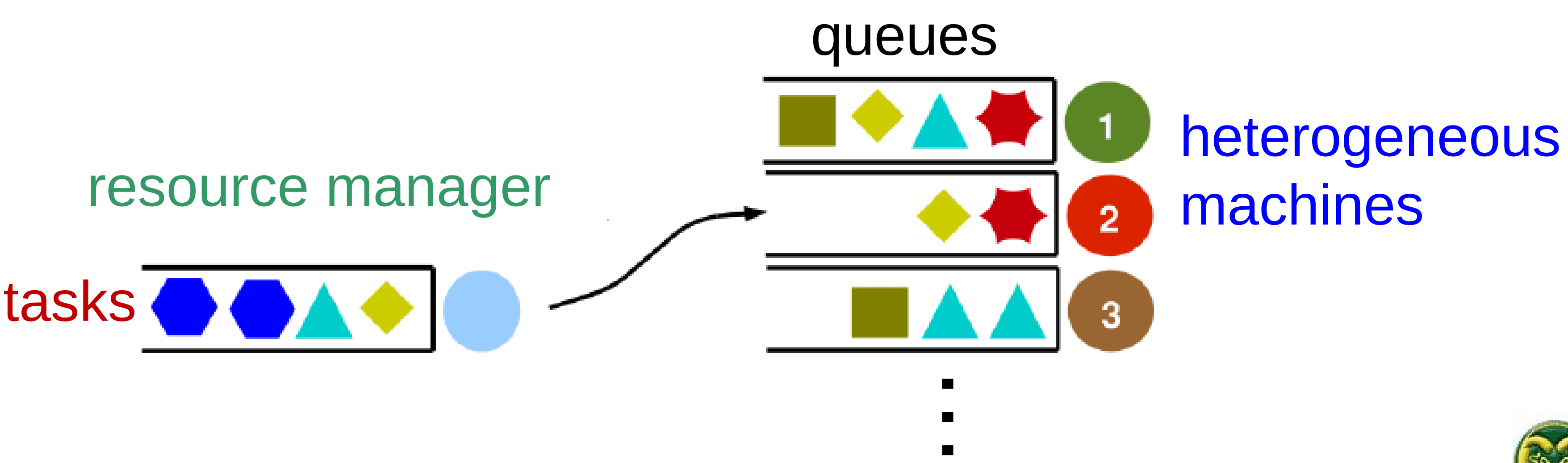
- *N* = 128 tasks, *M* = 8 machines, SRM value set to 90%
- 50 simulation trials, different PMFs for task/machine pairs
- 95% confidence intervals shown

Outline

- definition and stochastic model of robustness
- use in static resource allocation heuristics
- **use in dynamic resource allocation heuristics**
- summary and concluding remarks

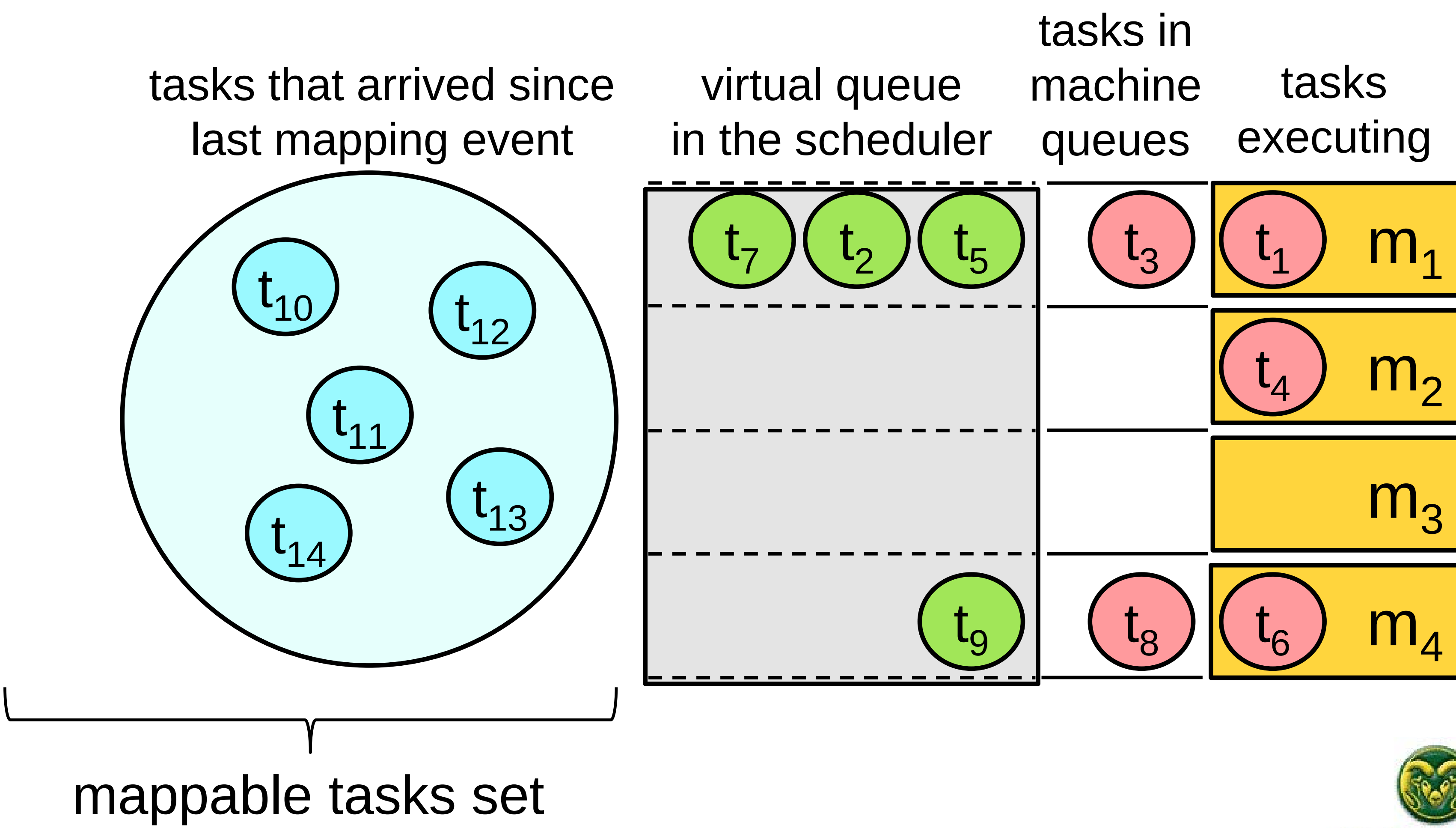
Problem Statement for Dynamic Resource Allocation

- cluster of *M* oversubscribed heterogeneous machines
- each dynamically arriving task has two elements
  - task type: stochastic execution time of the task (PMF)
  - deadline: for completing that **individual** task
- **goal:** maximize the number of tasks completed by their individual deadlines



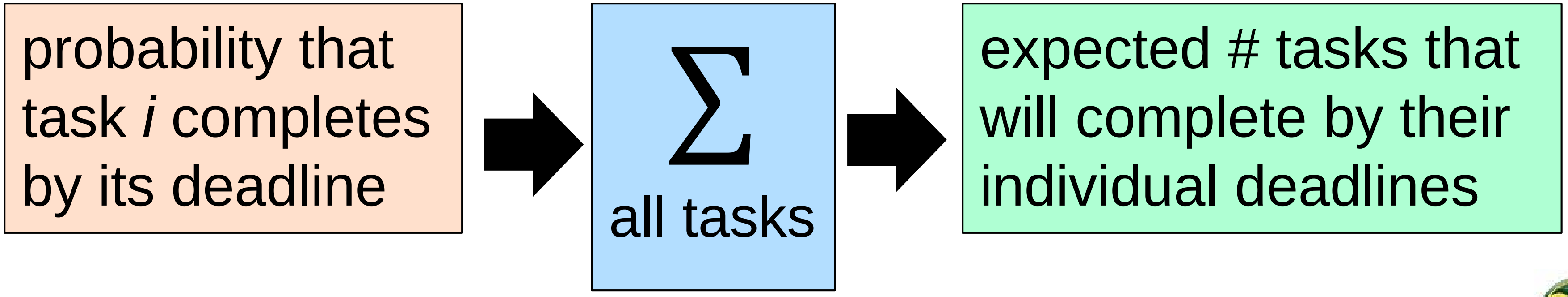
Mapping Event

- mapping event: when resource manager assigns to machines
- the batch of mappable tasks considered at an event

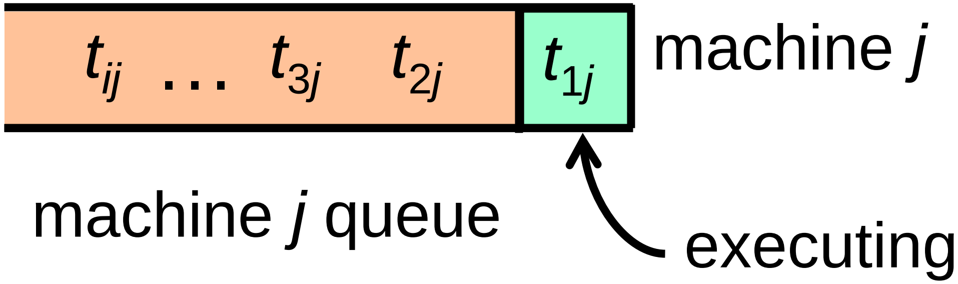


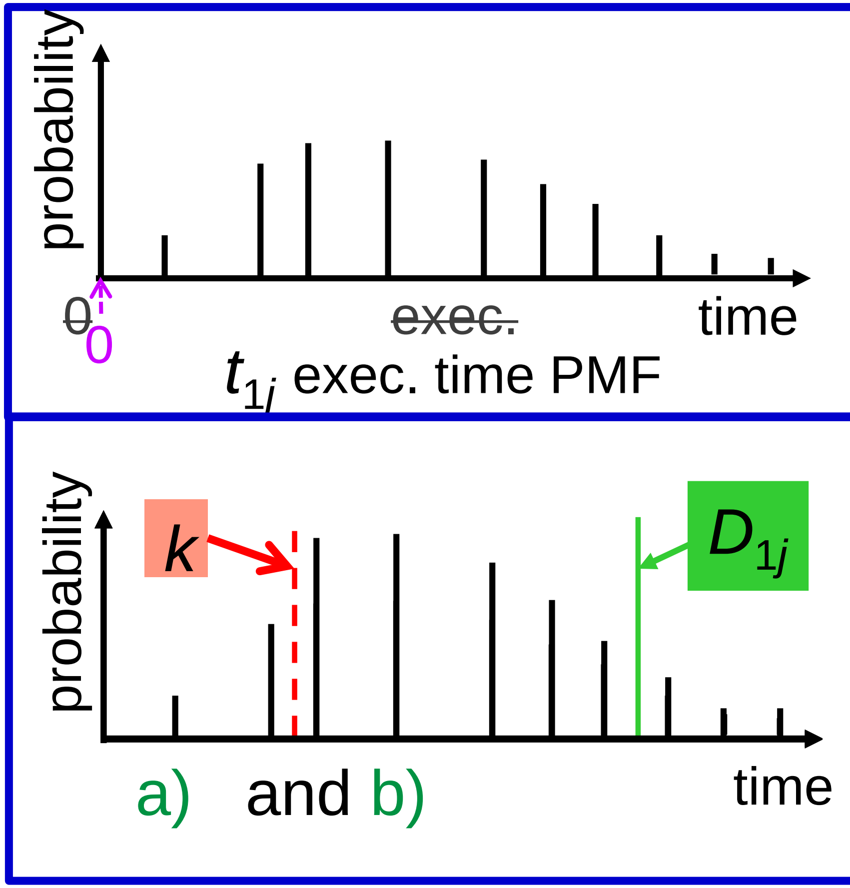
Robustness for Dynamic Resource Allocation

- what behavior makes the system robust?
  - completing all tasks by their individual deadlines
- what uncertainty is the system is robust against?
  - task execution times may vary substantially
- how is robustness of the system quantified?
  - expected number of queued and executing tasks that will complete by their individual deadlines



Probability Completing Executing Task by Deadline



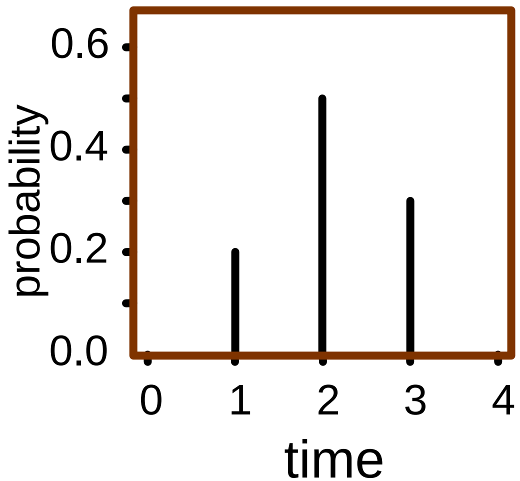


- new mapping event time *k*
- $\rho(t_{1j})$: probability of t_{1j} completing by its deadline
  - a) time *k* = current time
    - drop pulses < *k*
    - renormalize
  - b) sum pulses < deadline D_{1j}

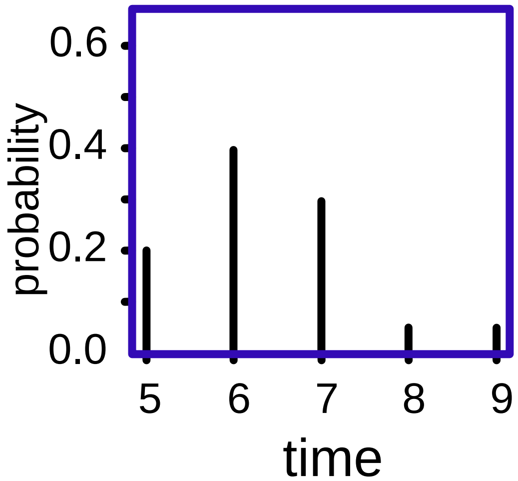
PMF for Completion Time of Task *i* for *i* > 1

- recall: t_{ij} is i^{th} task assigned to machine *j* at time *k*
- iterative procedure for finding completion time of t_{ij} for $i > 1$
- two cases for t_{ij} with deadline at, for example, time 8
  - executes on machine *j*
  - cannot start before deadline and is dropped

execution PMF for t_{ij}

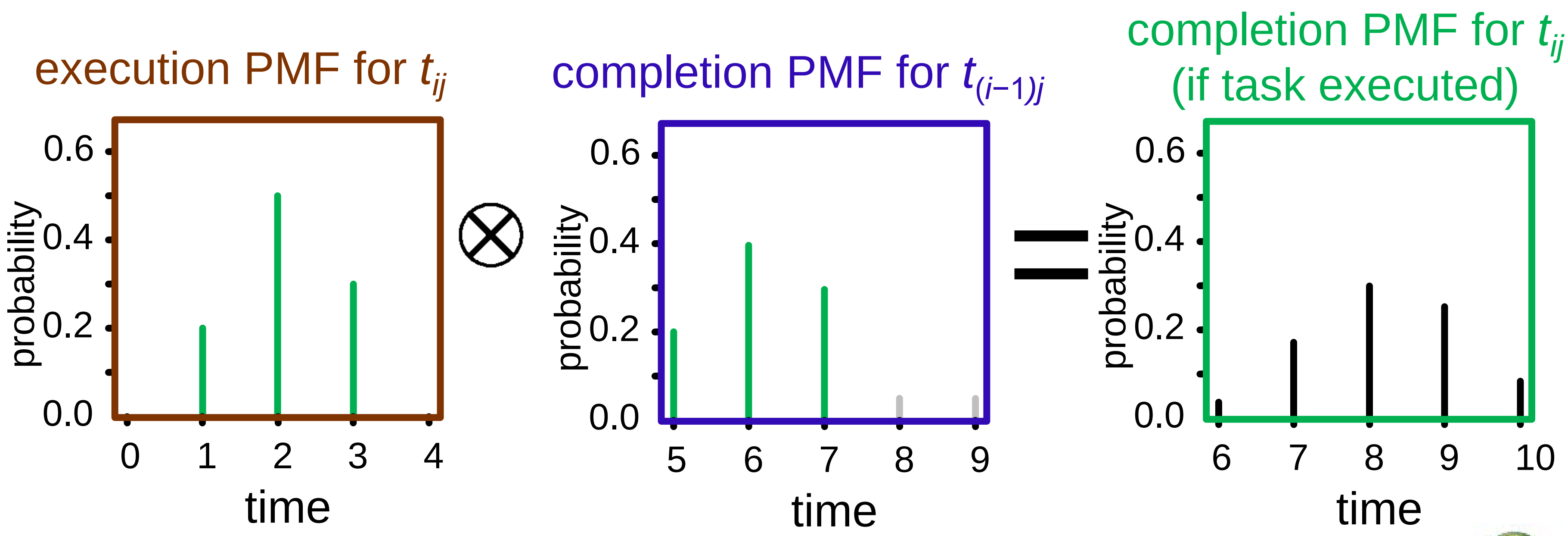


completion PMF for $t_{(i-1)j}$



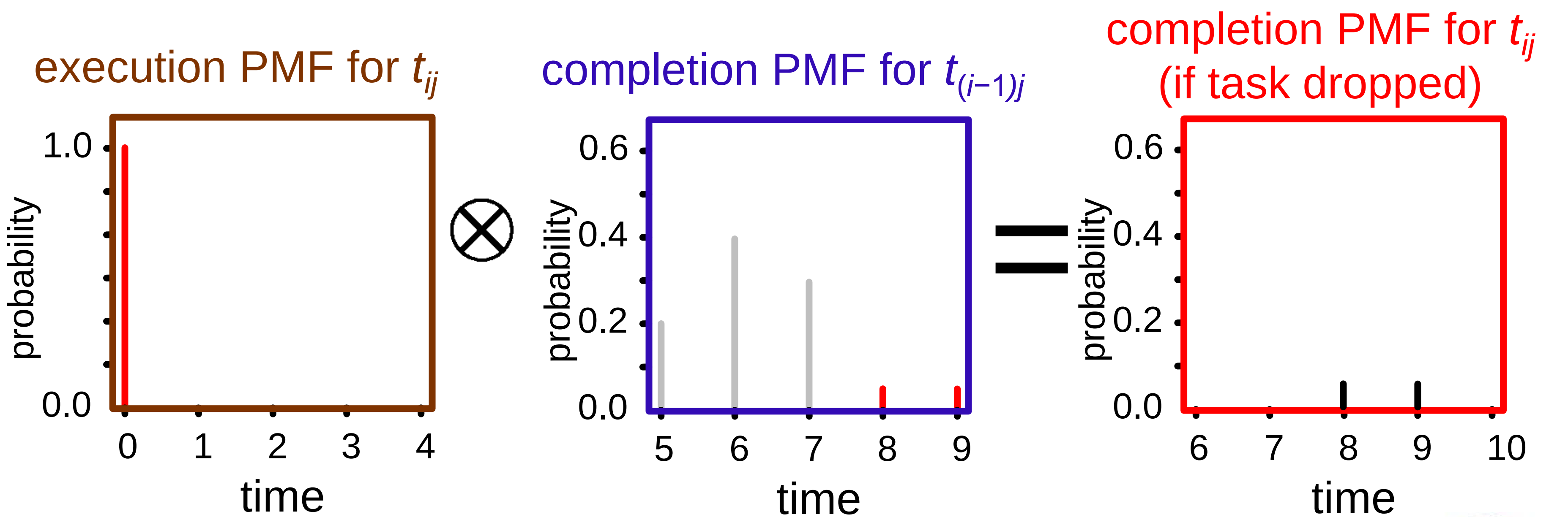
PMF for Completion Time of Task *i* for *i* > 1

- recall: t_{ij} is i^{th} task assigned to machine *j* at time *k*
- iterative procedure for finding completion time of t_{ij} for $i > 1$
- two cases for t_{ij} with deadline at, for example, time 8
  - executes on machine *j*
  - cannot start before deadline and is dropped



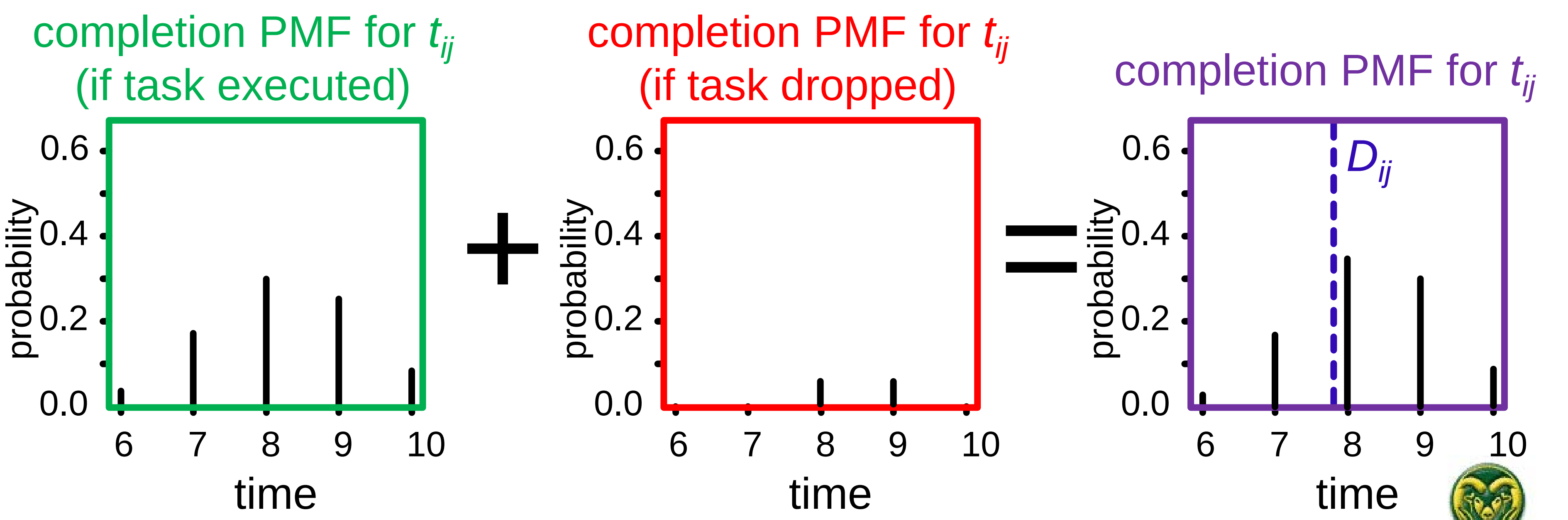
PMF for Completion Time of Task *i* for *i* > 1

- recall: t_{ij} is i th task assigned to machine j at time k
- iterative procedure for finding completion time of t_{ij} for $i > 1$
- two cases for t_{ij} with deadline at, for example, time 8
  - executes on machine j
  - cannot start before deadline and is dropped



PMF for Completion Time of Task *i* for *i* > 1

- recall: t_{ij} is i^{th} task assigned to machine *j* at time *k*
- iterative procedure for finding completion time of t_{ij} for $i > 1$
- two cases for t_{ij} with deadline at, for example, time 8
  - executes on machine *j*
  - cannot start before deadline and is dropped
- sum pulses < deadline D_{ij} to get $\rho\left(t_{ij}\right)$



Stochastic Robustness for Dynamic Heuristics

$\rho\left(t_{ij}\right)$

probability that task t_{ij} completes before its deadline

$\sum_i \rho\left(t_{ij}\right)$

$\rho_j^{(k)}$

expected number of tasks completed by machine j before their deadlines measured at time k

recall: t_{ij} is i th task assigned to machine j at time k

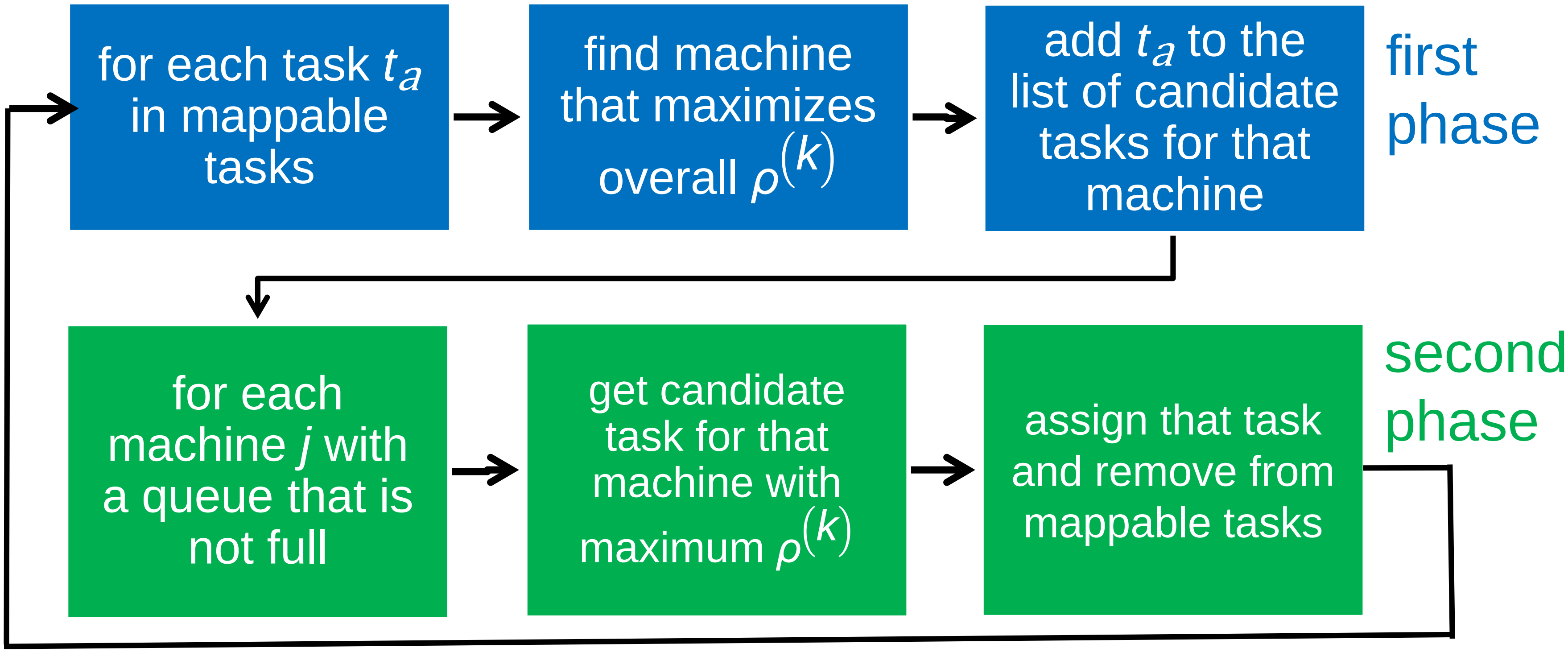
$\sum_j \rho_j^{(k)}$

stochastic dynamic robustness: $\rho^{(k)}$
the expected number of tasks that will meet their deadlines measured at time k

Colorado State University

Heuristic: Maximum On-time Completions (MOC)

- during a mapping event at time k



Comparison Heuristics

- **Heuristic: Min Completion - Min Completion (MM)**

phase 1: for each of the mappable tasks find machine with minimum expected completion time

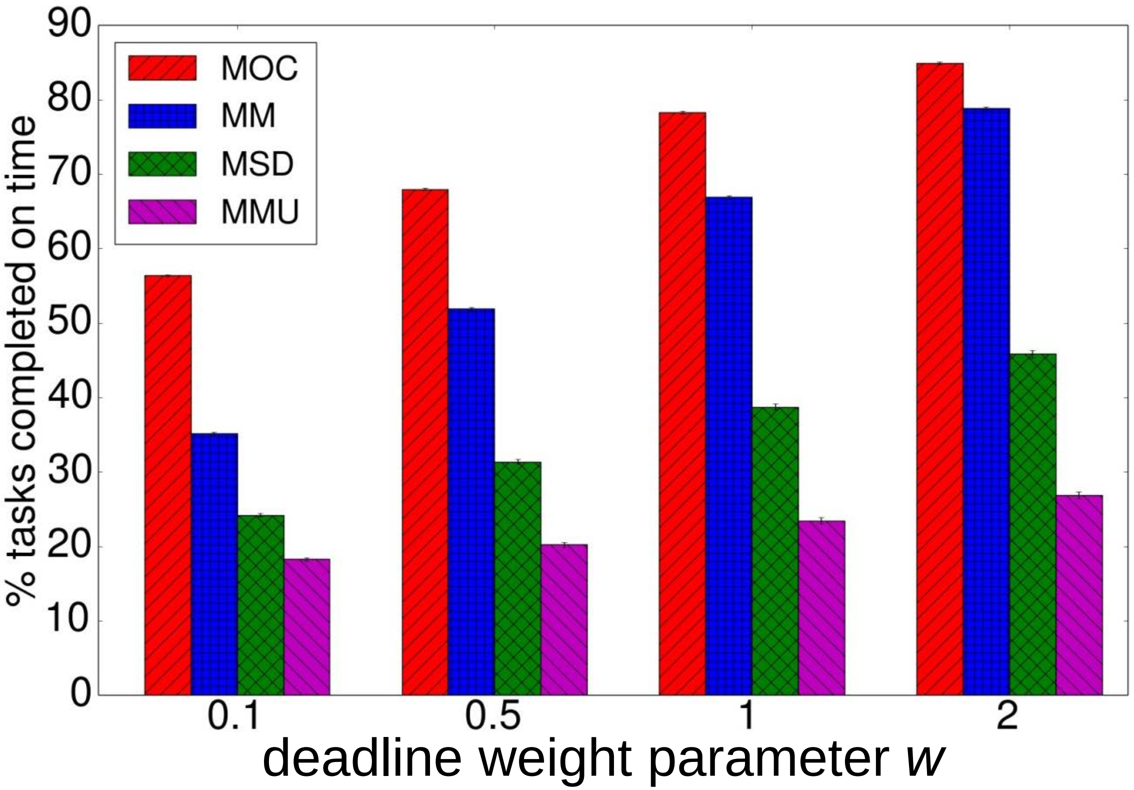
phase 2: provisionally map task in task/machine pair with the minimum expected completion time

  - move provisionally mapped tasks to machine queues until full

- **Heuristic: Min Completion - <u>Max Urgency (MMU)</u>**
  - **phase 2:** map task in task/machine pair that <u>maximizes</u> <u>urgency = 1/(task deadline − expected completion time)</u>

- **Heuristic: Min Completion - <u>Soonest Deadline (MSD)</u>**
  - **phase 2:** map task in task/machine pair with the <u>soonest deadline</u>

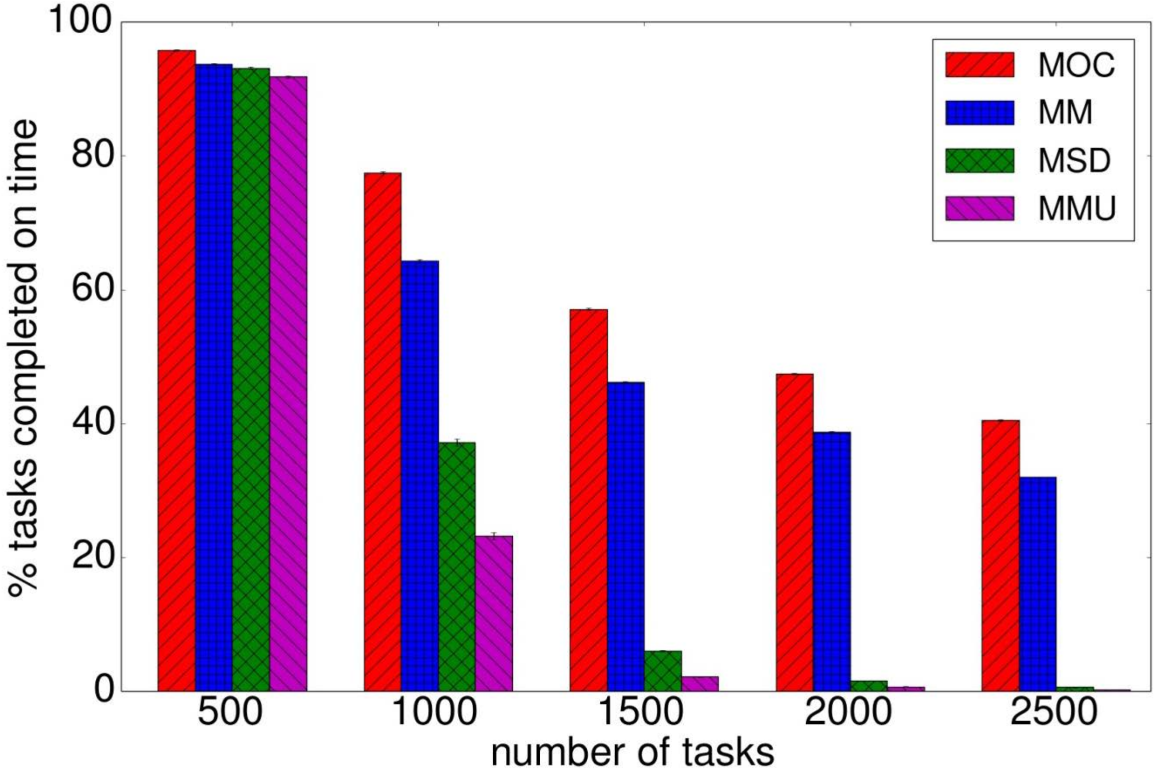
Results: Varied Deadline Weight Parameter (*w*)



- 1,000 tasks
- 8 machines
- queue size 2
- 100 simulation trials
  - task types, task arrivals
- 95% confidence intervals
- MOC: Max On-time Comp.
- MM: Min Comp. - Min Comp.
- MSD: Min Comp - Soonest Deadline
- MMU: Min Comp. - Max Urgency

- deadline for task t_i = t_i arrival time + average t_i exec. time + w × (average exec. time over all tasks)
- problem is harder with tighter deadlines (smaller *w*)
- MOC best performing heuristic - uses stochastic robustness

Results: Varied Number of Tasks in Workload

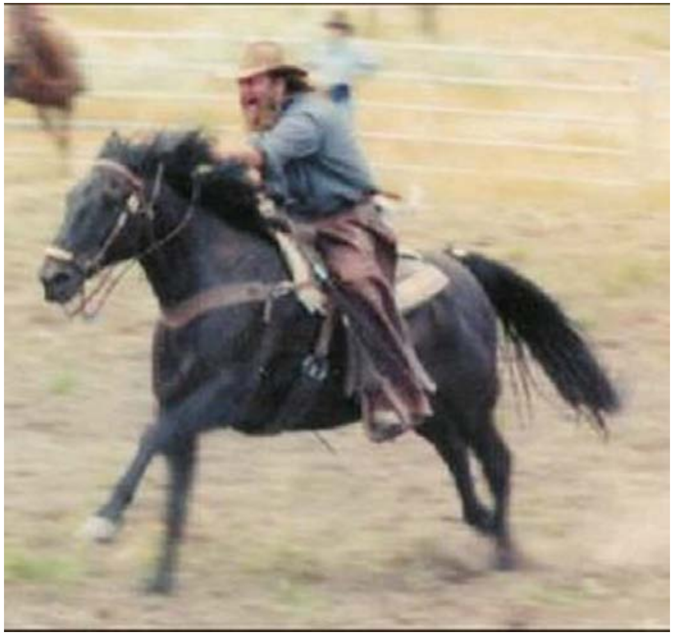


- deadline weight $w = 1$
- 8 machines
- queue size 2
- 100 simulation trials
  - task types, task arrivals
- 95% confidence intervals
- MOC: Max On-time Comp.
- MM: Min Comp. - Min Comp.
- MSD: Min Comp - Soonest Deadline
- MMU: Min Comp. - Max Urgency

- MOC best because tried to maximized $\rho^{(k)}$ robustness
- MM second best because attempted to min. execution time
- MMU and MSD perform worse because they choose tasks with a high probability to miss their deadlines

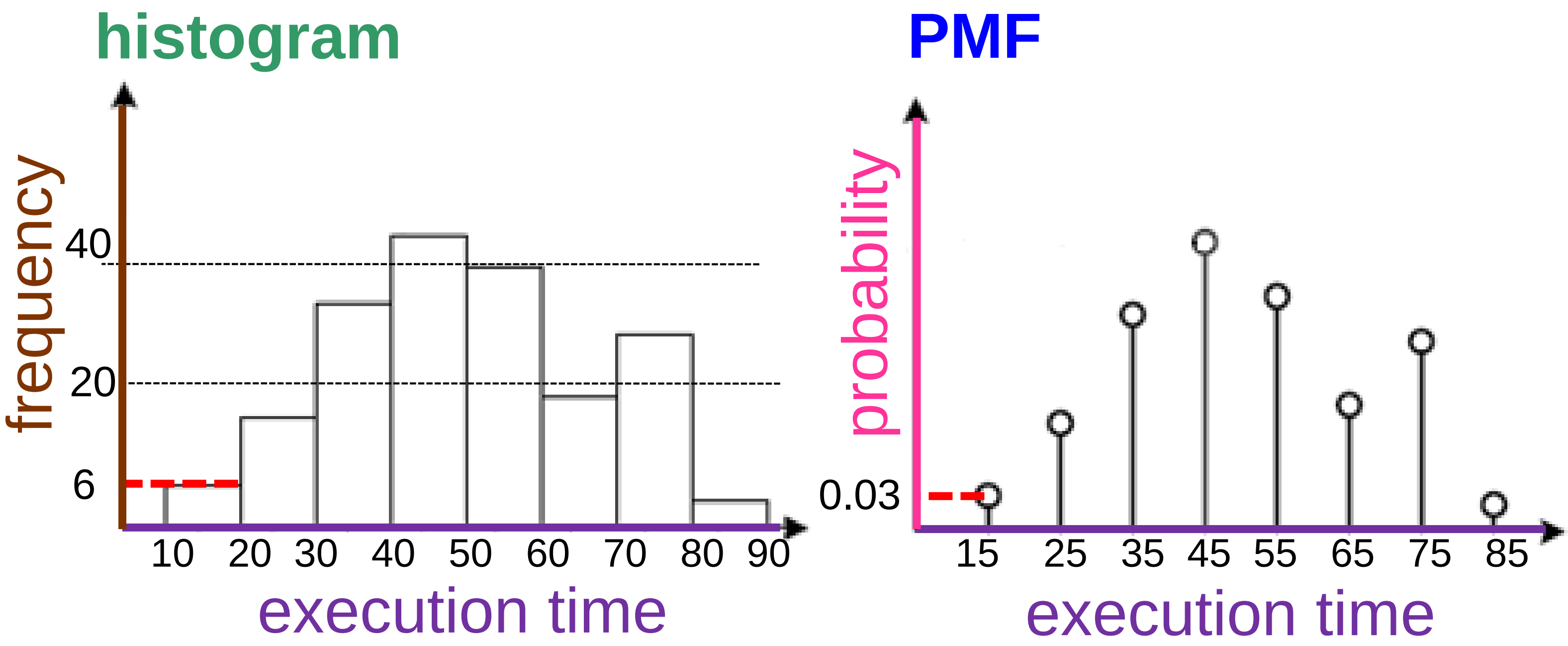
Outline

- definition and stochastic model of robustness
- use in static resource allocation heuristics
- use in dynamic resource allocation heuristics
- **summary and concluding remarks**



Summary

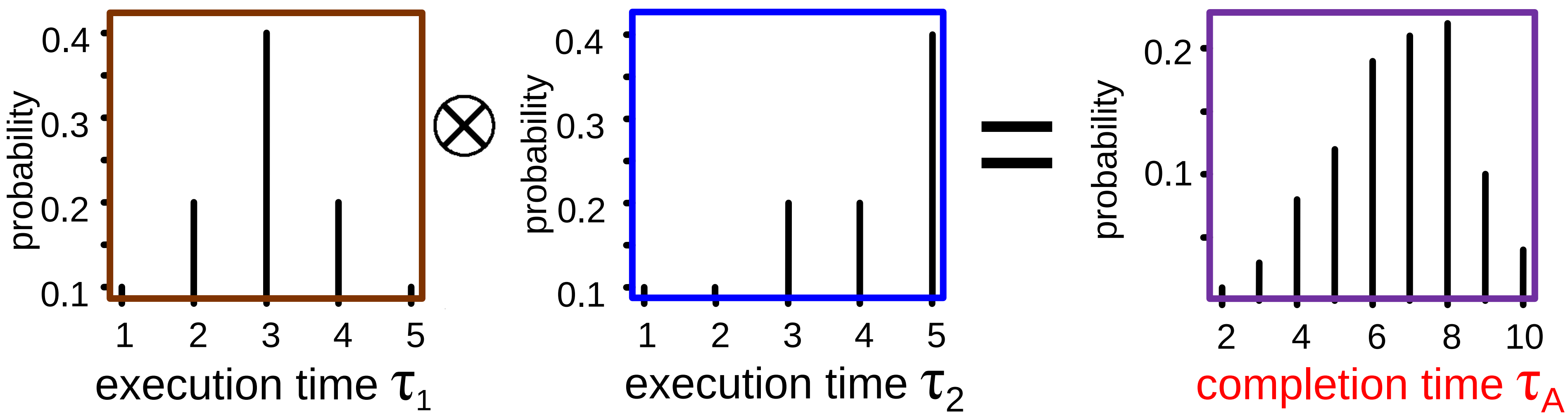
1) build histogram and convert to probability mass function (PMF)



Summary

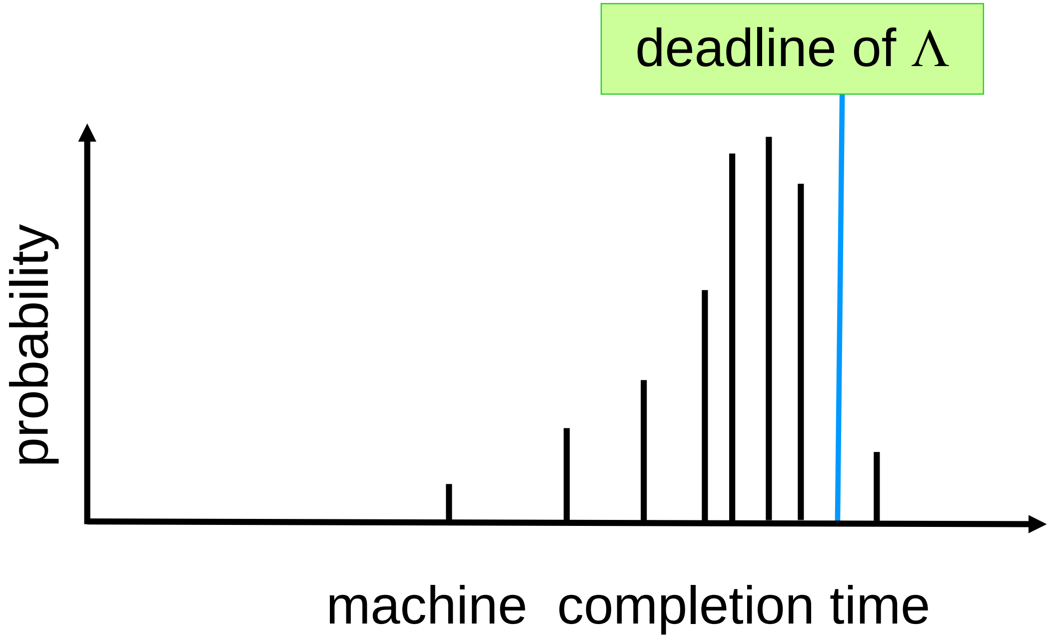
1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs

PMF for t_1 on machine A ⊗ PMF for t_2 on machine A = PMF for completion time of machine A



Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline



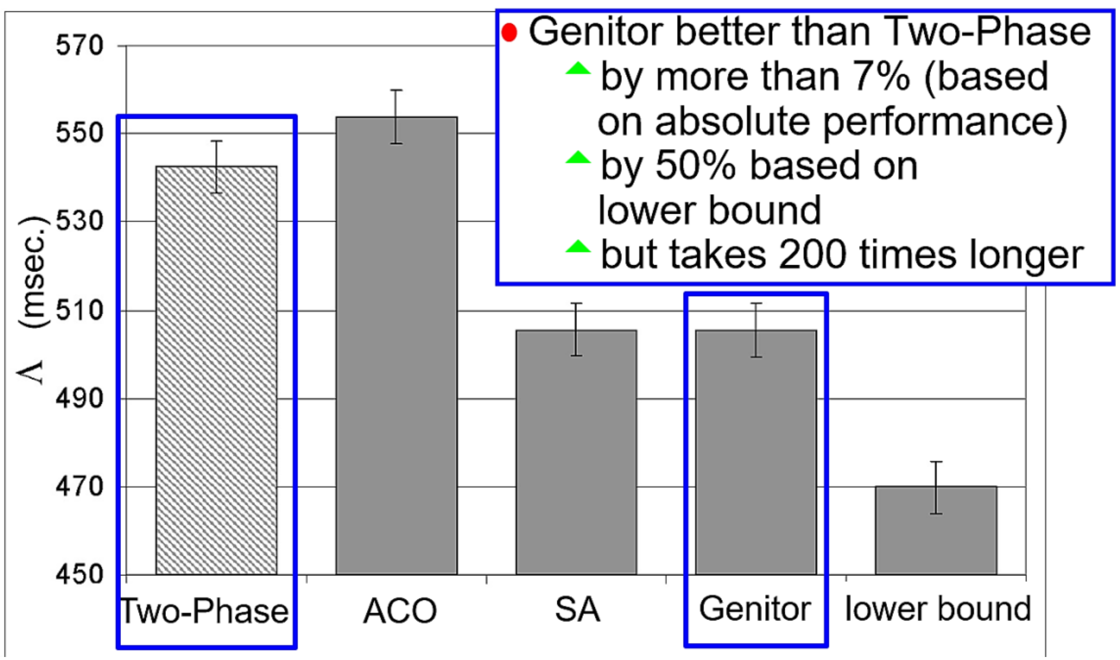
Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)

$$\prod_{j=1}^{M} \text{Prob}[S_j \leq \Lambda]$$

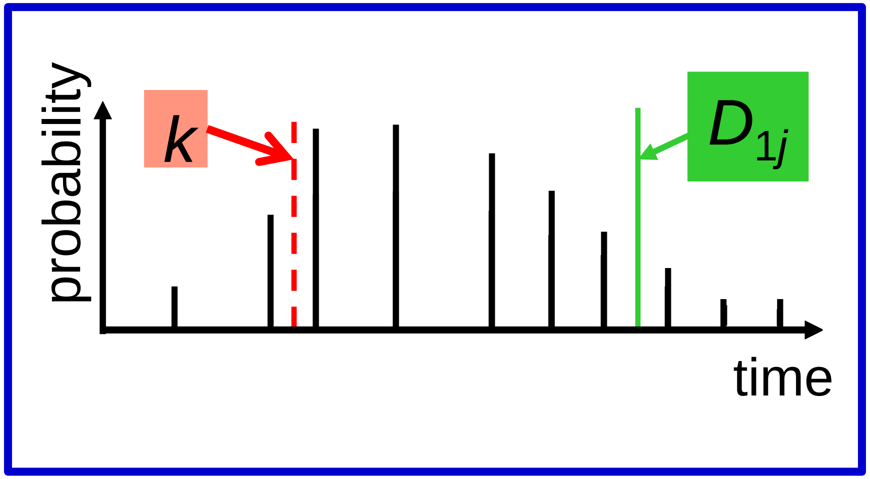
Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)
5) use SRM in static resource allocation heuristics



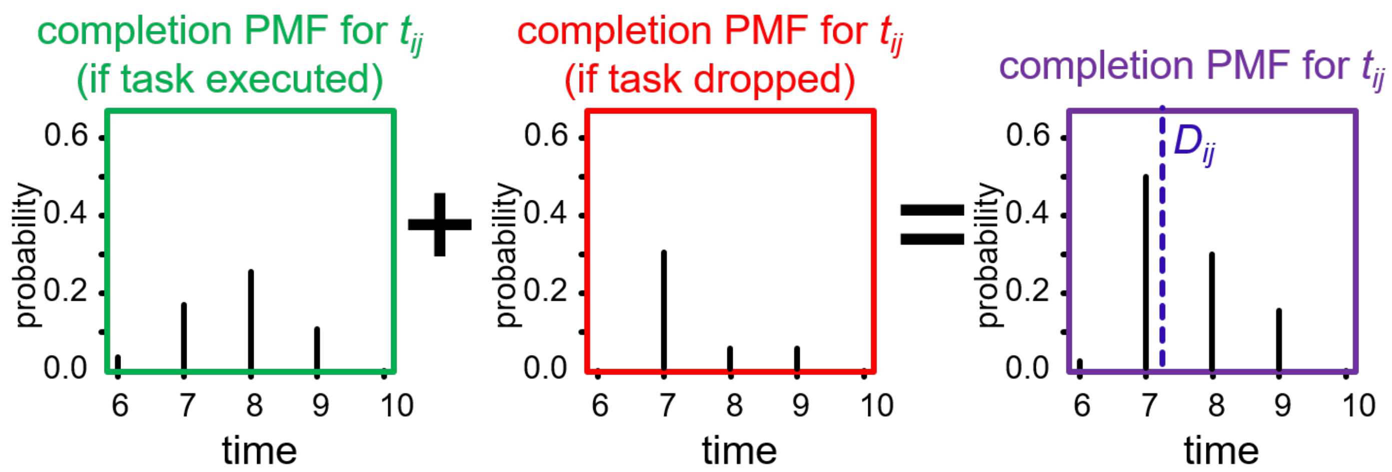
Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)
5) use SRM in static resource allocation heuristics
6) probability completing executing task by individual deadline



Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)
5) use SRM in static resource allocation heuristics
6) probability completing executing task by individual deadline
7) probability completing task i+1 by individual deadline



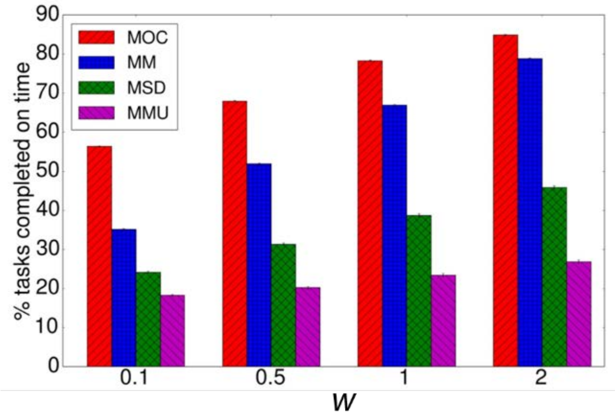
Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMFs
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)
5) use SRM in static resource allocation heuristics
6) probability completing executing task by individual deadline
7) probability completing task *i*+1 by individual deadline
8) robustness = expected # tasks meet individual deadlines

$$\sum_{ij} \rho\left(t_{ij}\right)$$

Summary

1) build histogram and convert to probability mass function (PMF)
2) task execution time PMFs to machine completion time PMF
3) probability given machine will meet common task deadline
4) probability all machines will meet common task deadline (SRM)
5) use SRM in static resource allocation heuristics
6) probability completing executing task by individual deadline
7) probability completing task *i*+1 by individual deadline
8) robustness = expected # tasks meet individual deadlines
9) use this robustness in dynamic resource allocation heuristic



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?
- work on robust resource allocation problems
  - publish papers about your work!
- thank you for listening
  - **The End**

References & Sponsors for Our Research Presented

- definition and stochastic model of robustness
  - J. Smith et al., “Robust Resource Allocation in Heterogeneous Parallel and Distributed Computing Systems,” in *Wiley Encyclopedia of Computing*, 2008
- use in static resource allocation heuristics
  - V. Shestak et al., “Stochastic Robustness Metric and its Use for Static Resource Allocations,” *Journal of Parallel & Distributed Computing*, Aug. 2008
- use in dynamic resource allocation heuristics
  - M. Salehi et al., “Stochastic-based Robust Dynamic Resource Allocation for Independent Tasks in a Heterogeneous Computing System,” *Journal of Parallel & Distributed Computing*, Nov. 2016
- sponsors of this research