Robust Computing Systems

"Sanitized" version of DataSys 2012 Keynote slides presented on Wednesday, October 24, 2012,

by

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

for posting on the conference website

Robust 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

- definition of robustness
- stochastic model and metric for robustness
- integration into static resource allocation heuristics
- use of model for a dynamic environment
- conclusions

Brief Self-Introduction

- Professor, Colorado State University, USA
- BS degree: MIT
- Ph.D. degree: Princeton
- Fellow IEEE
- Fellow ACM

Applicability of Stochastic Robustness Model

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

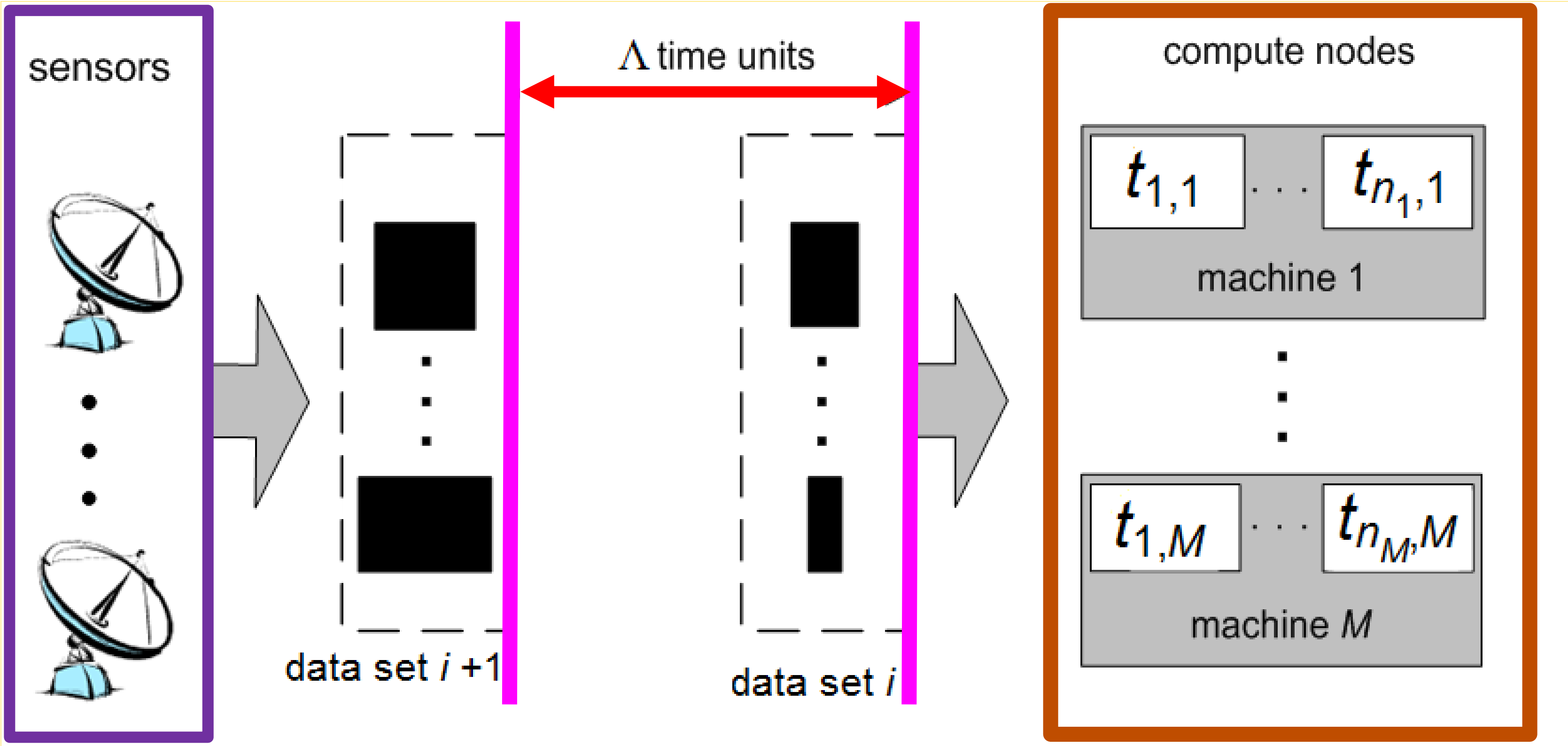
Heterogeneous Parallel Computing System

- 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 allocation**:
  assign (map) tasks to machines
  to optimize some performance measure
  - NP-complete (cannot find optimal in reasonable time)
  - ex.: 5 machines and 30 tasks → 5^{30} possible assignments
    - 5^{30} nanoseconds > 1,000 years!
  - use heuristics to find near optimal allocation

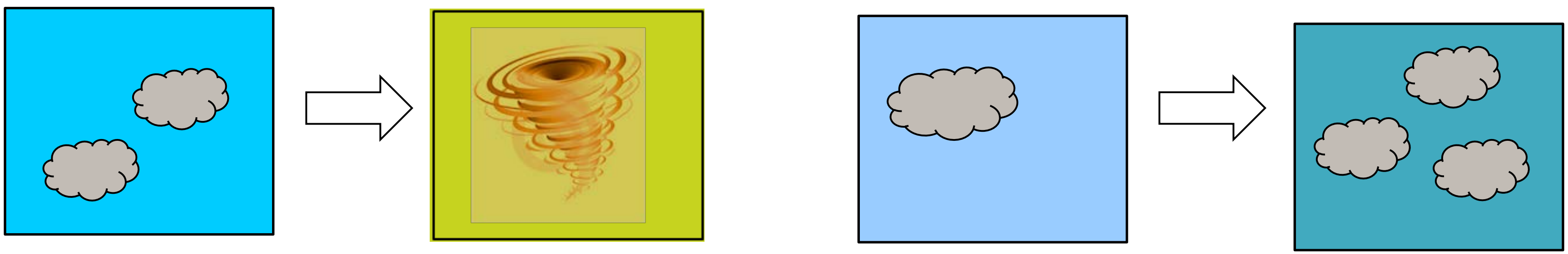
Ex.: Radar Data Processing for Weather Forecasting



- sensors produce periodic data sets, each with multiple data files
- ***N*** independent tasks process each data set within **Λ** time units
- *N* tasks **statically** assigned to ***M*** heterogeneous machines, *N*>*M*
- similar computing environments
  - satellite data sets for producing maps
  - surveillance data sets for homeland security

Uncertainty in Environment

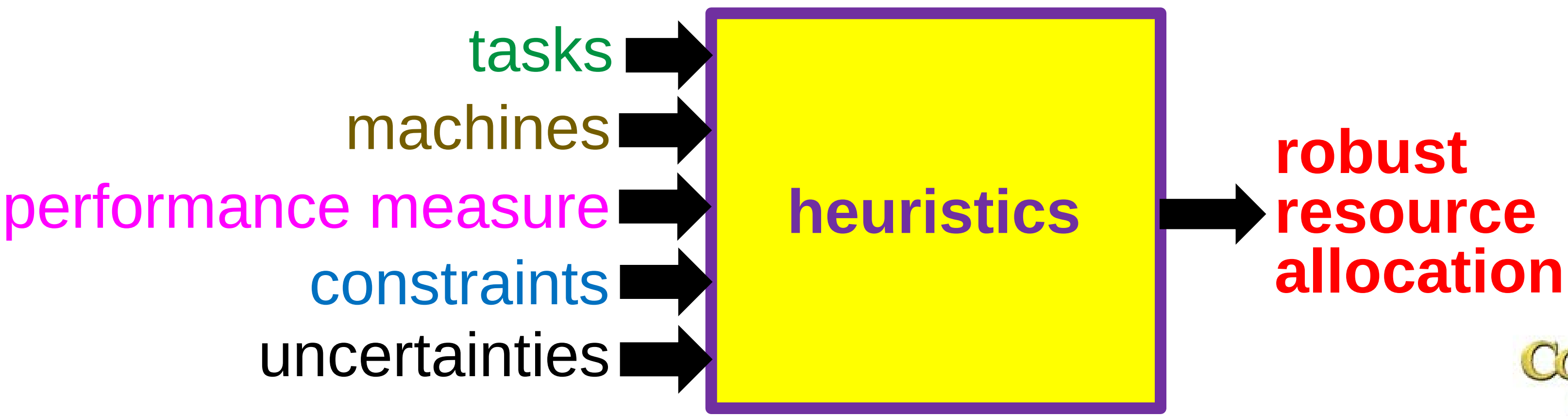
- **variability** across the data sets results in variability of the execution time of each task even on the same machine
  - examples
    - types of objects found in a radar scan data file
    - increase in number of objects in a radar scan data file



- unable to predict exact execution times of tasks
  - **uncertainty** parameters in the system
  - know **history** of task execution times on each machine over different data sets
- need to find resource allocation of tasks to machines that is **robust** against this uncertainty by using this history

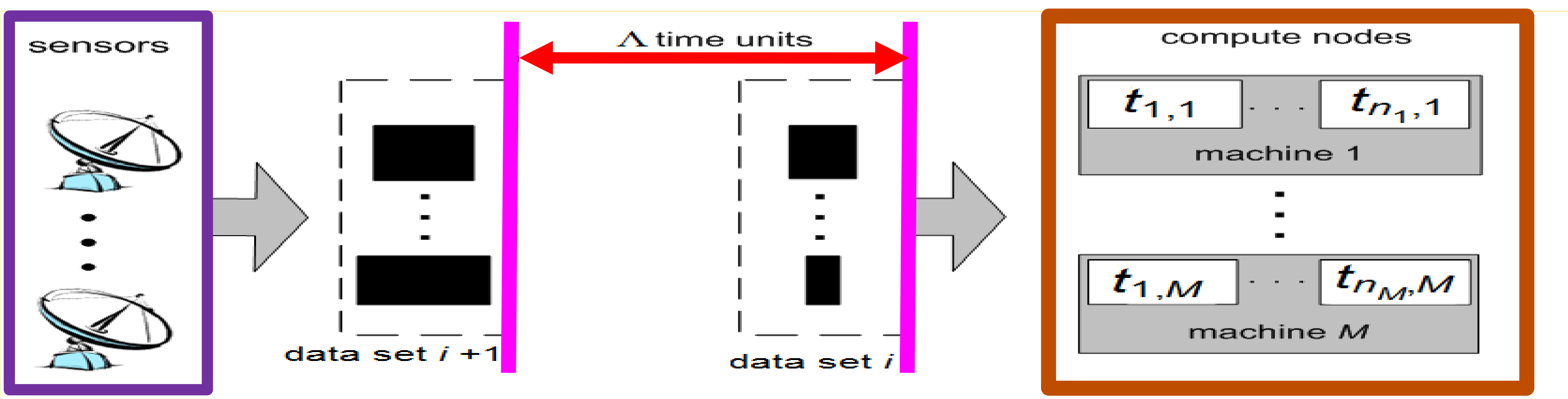
Problem Statement

- unpredictable execution times of the tasks across data sets
- calculate the probability that every data set is processed before the next data set arrives
  - have a probabilistic guarantee of performance
- <u>**problem statement**</u>
  - determine a **robust** static resource allocation
  - 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 Resource Allocation

- term "robustness" usually used without explicit definition
- **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

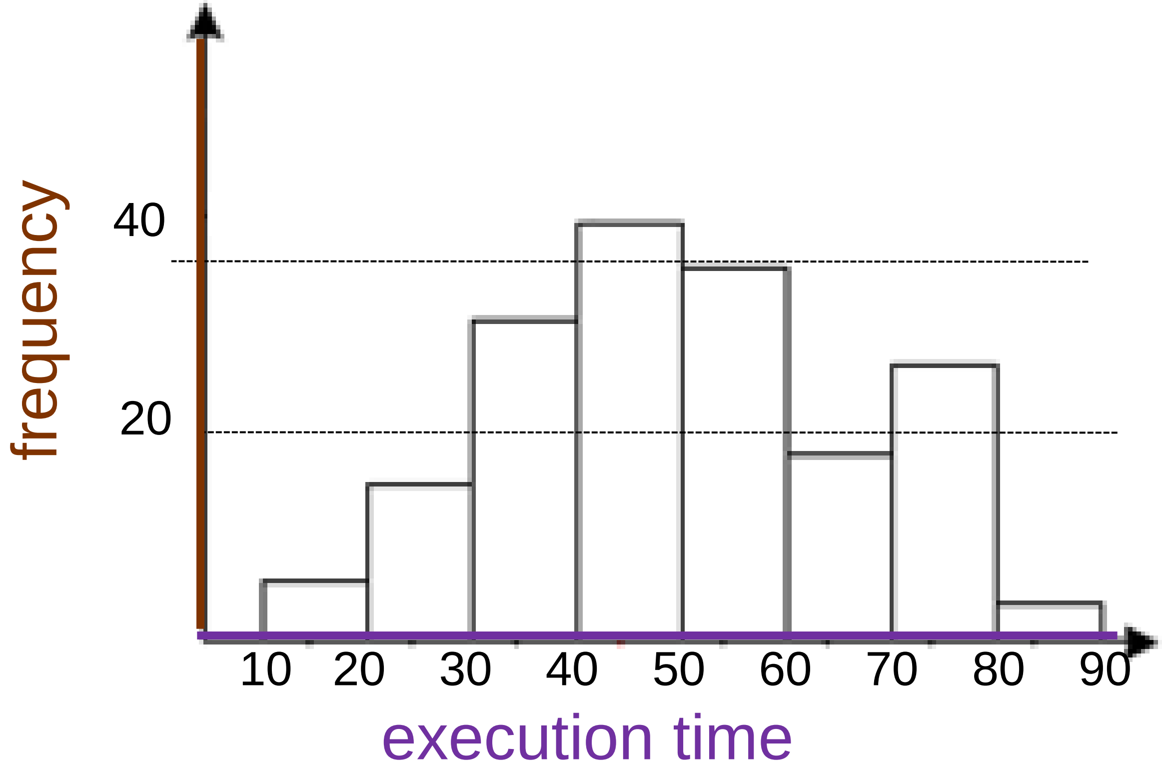


Outline

- definition of robustness
- stochastic model and metric for robustness
- integration into static resource allocation heuristics
- use of model for a dynamic environment
- conclusions

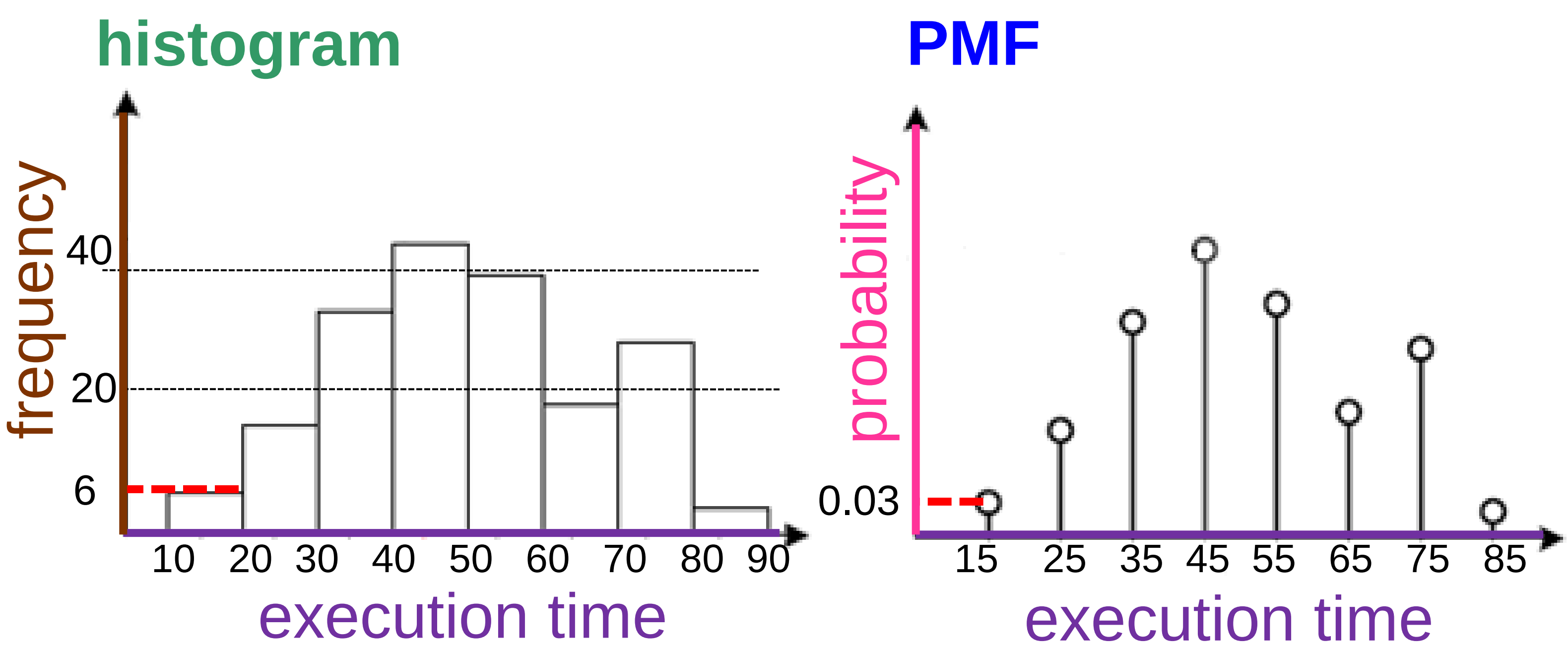
Construct Histogram from Collected Information

- know **history** of task execution times on each machine over different data sets
- consider collecting samples of how long a given task has taken to execute on a given machine in a **histogram**
  - 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%



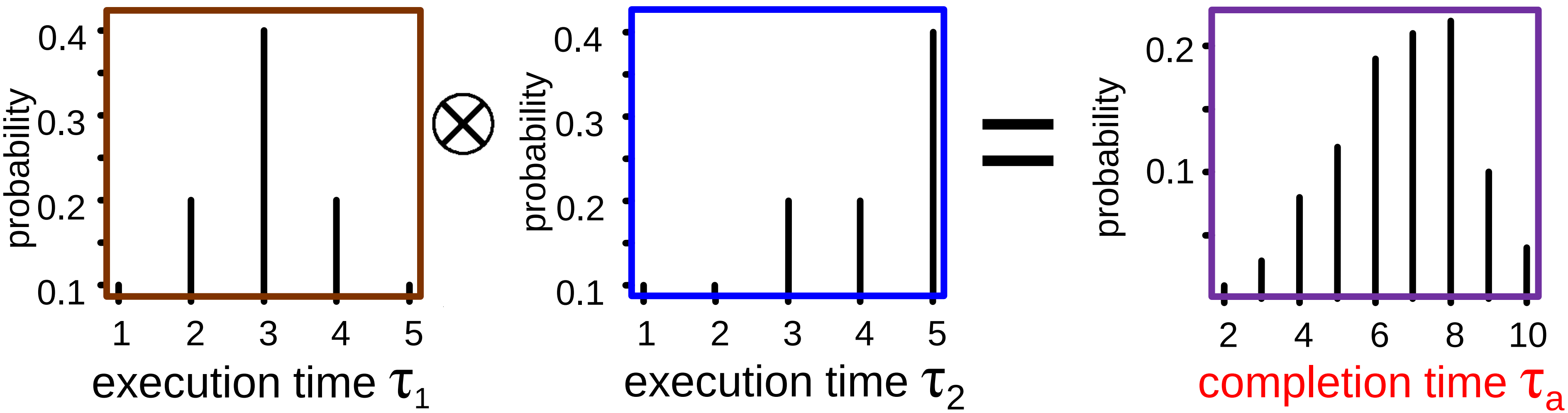
PMF for Completion Time of Machine

- 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

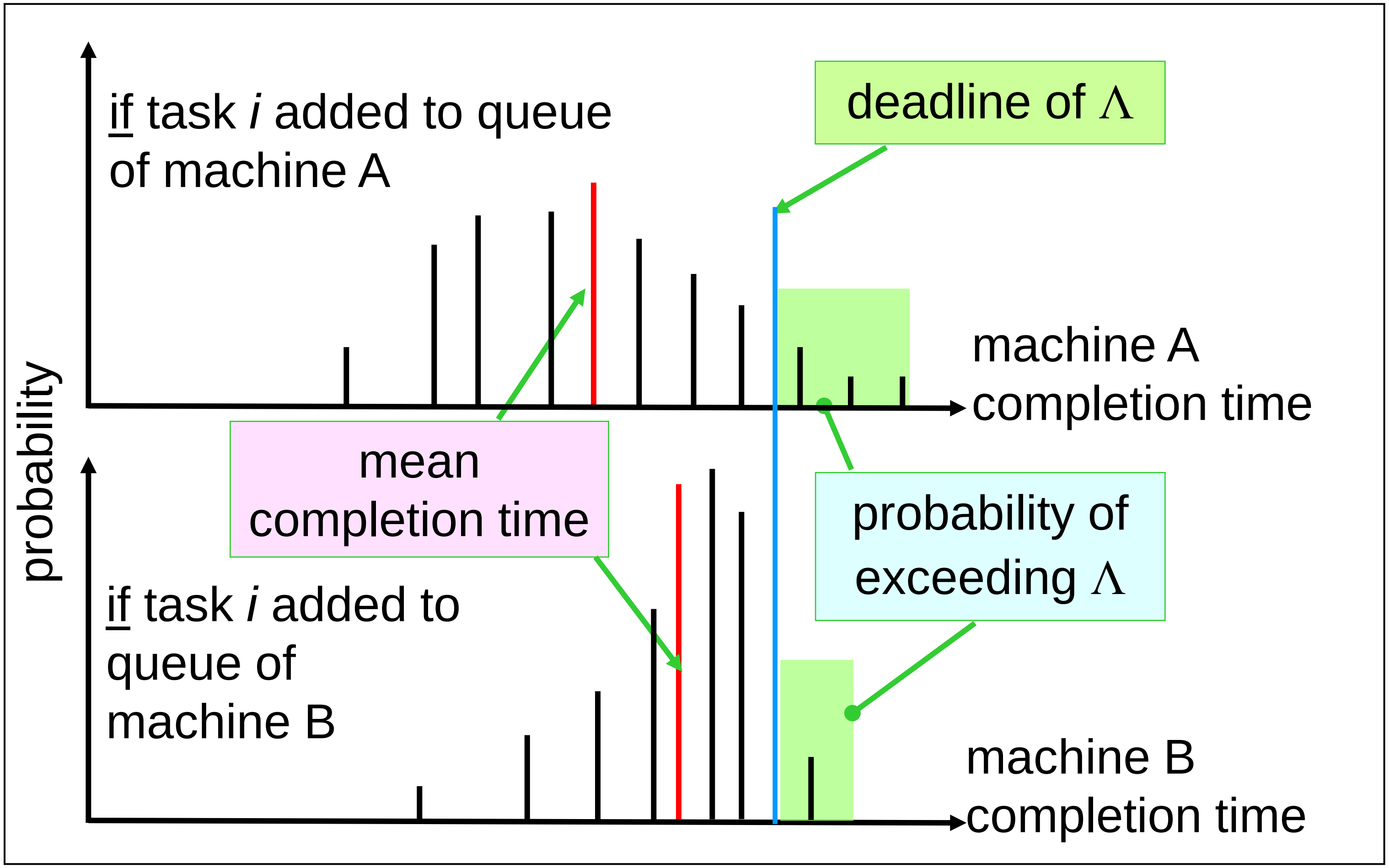
PMF for completion time of machine A



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

Intuitive View of Stochastic Robustness

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



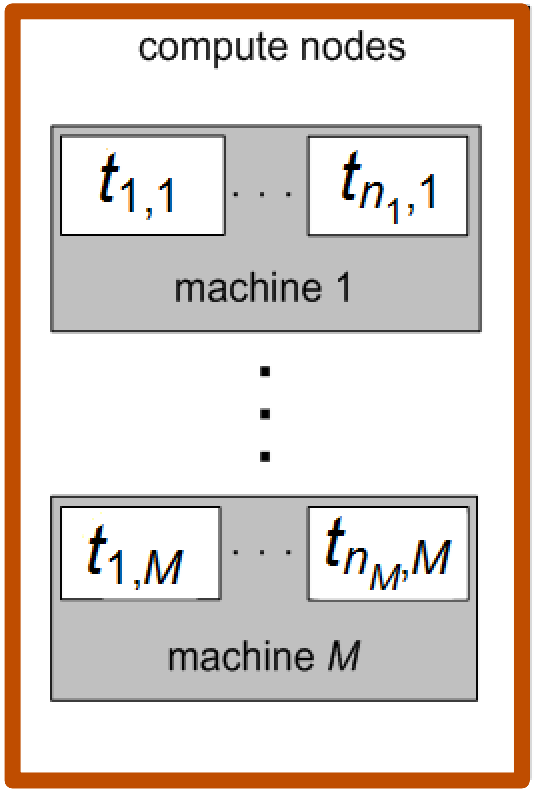
- assign task *i* to machine A or B?
- mean → A
- sum of heights of pulses > deadline → B

Stochastic Robustness Heuristic Goals

- T_{ij} : execution time random variable for task *i* on machine *j*
- S_j : stochastic completion time for machine *j* (tasks independent) $S_j = \sum_{i=1}^{n_j} T_{ij}$
- Λ : deadline for completing all tasks
- machine *j* stochastic robustness Prob[$S_j \leq \Lambda$]
- **Stochastic Robustness Metric (SRM)**
  - assuming independence of machines

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

- **goal of heuristics** – two possible robustness situations
  - maximize <u>SRM</u> for a given <u>Λ</u> value
  - minimize <u>Λ</u> for a given <u>SRM</u> value



Outline

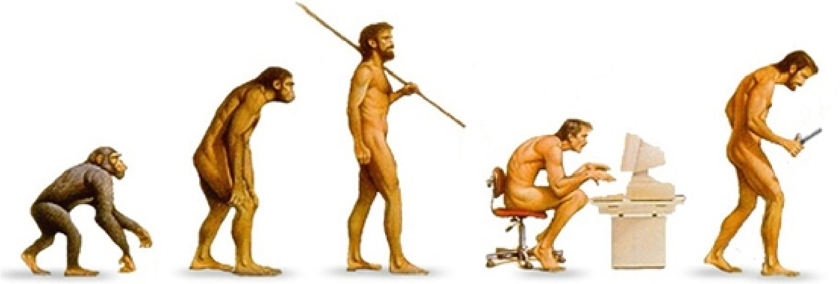
- definition of robustness
- stochastic model and metric for robustness
- integration into static resource allocation heuristics
- use of model for a dynamic environment
- conclusions

Colorado State University

Static Resource Allocation Heuristics

- goal: static assignment of *N* tasks to *M* machines
  - minimize Λ for a given SRM value, for example 90%
- greedy heuristics
  - example: Two-Phase
  - allocation made with locally optimal decisions

- global heuristics
  - example: Genitor – steady-state genetic (evolutionary) algorithm
  - improve allocation over iterations



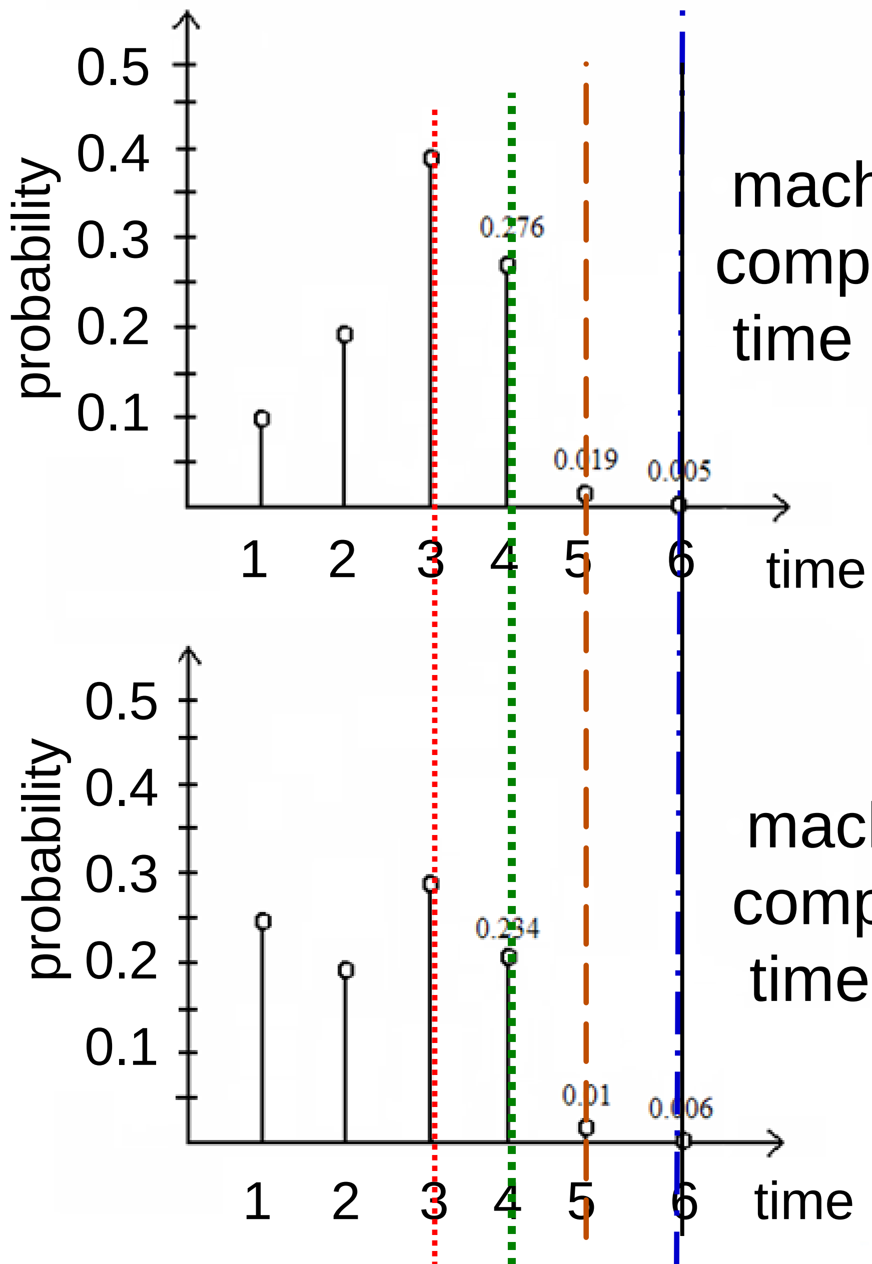
- greedy heuristic generally derives allocation faster than global
- global heuristics can improve upon greedy results
- use Lambda Minimization Routine (LMR) to find for a <u>given resource allocation</u> the minimum Λ is for SRM value of 90%

Lambda Minimization Routine (LMR)

- determines minimum Λ
  - for a given resource allocation
  - at required SRM (probability)
  - $$\prod_{j=1}^{M} \text{Prob}[S_j \le \Lambda] \ge 90\%$$
- ex.: SRM = 90%

SRM = 1 (Λ = 6)
0.97 (Λ = 5)
0.96 (Λ = 4)
0.525 (Λ = 3)

therefore
Λ = 4



machine A completion time PMF

machine B completion time PMF

Two-Phase Greedy Heuristic

- based on the concept of the Min-min heuristic
- $\Lambda(t_i,m_j)$ call to LMR function for minimum Λ **if** task t_i is added to machine m_j

Two-Phase Greedy procedure

- **while** there are still unmapped tasks
  - **phase 1:** for each of the unmapped tasks
    - j value that minimizes $\Lambda(t_i,m_j)$, $1 \le j \le M$
  - **phase 2:** among these task/machine pairs
    - find pair with minimum $\Lambda(t_i,m_j)$
    - map this task to its associated machine

Genitor Steady State Genetic Algorithm (GA)

- 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)

Genitor
Population: 200

- initial population generation
  - one chromosome: solution from the Two-Phase Greedy heuristic ("seed")
  - other 199: simple greedy heuristic

- population is put in ascending order based on minimum Λ value for the given SRM (probability)
  - LMR (Lambda Minimization Routine) is used to find minimum Λ value

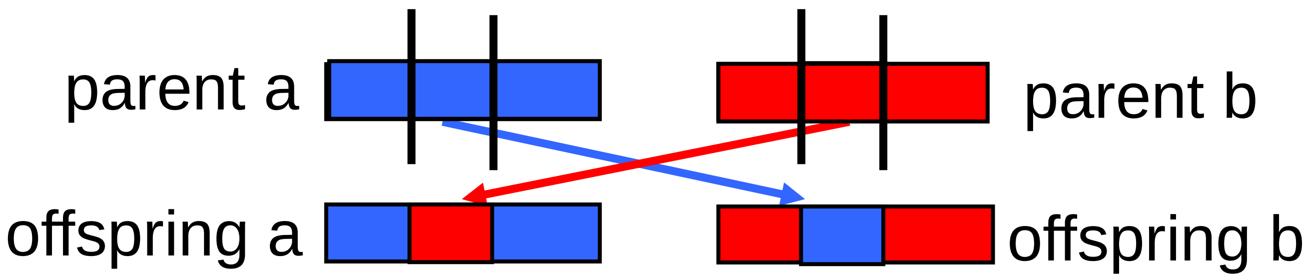
Procedure for Genitor

- generate initial population
- **while** stopping criterion
  - select two parent chromosomes from the 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

Colorado State University

Genitor: Crossover

- selection of parents is done probabilistically
- crossover is “two point reduced surrogate”
  - crossover points are randomly selected
    so that at least one element is different
  - elements between crossover points are exchanged
  - generates two offspring



Genitor: Mutation

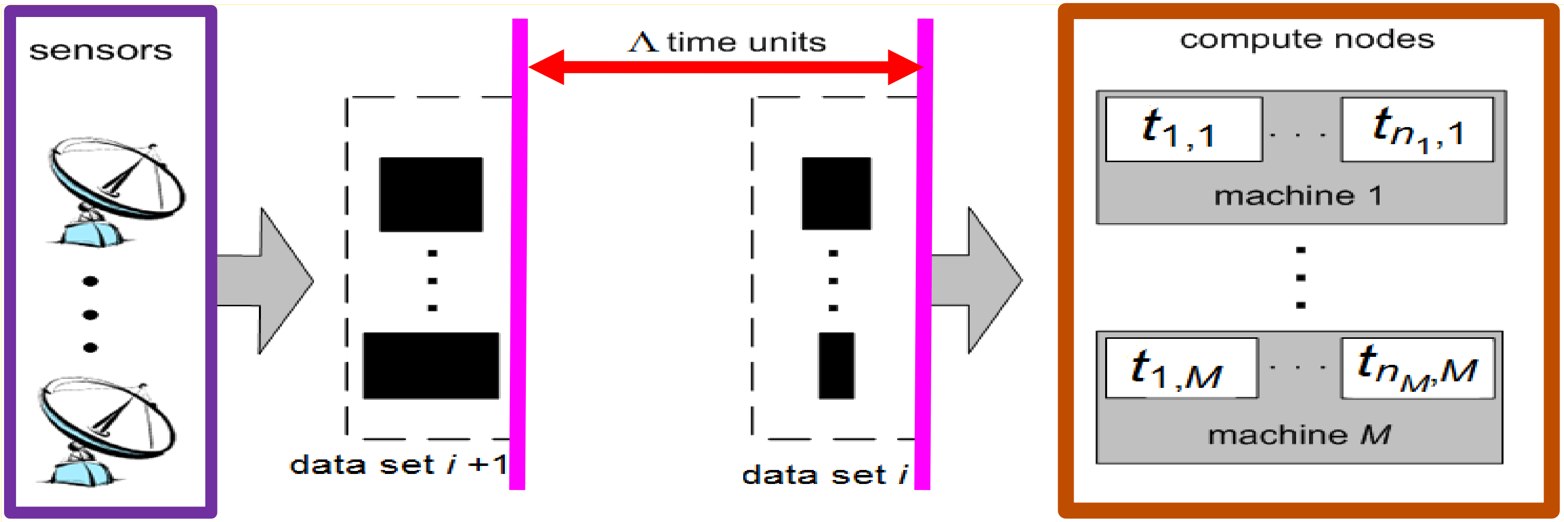
1	2	3	4	5	6	7	8	9	10	...
2	1	2	3	1	2	3	1	2	2	...

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

Genitor: Local Search

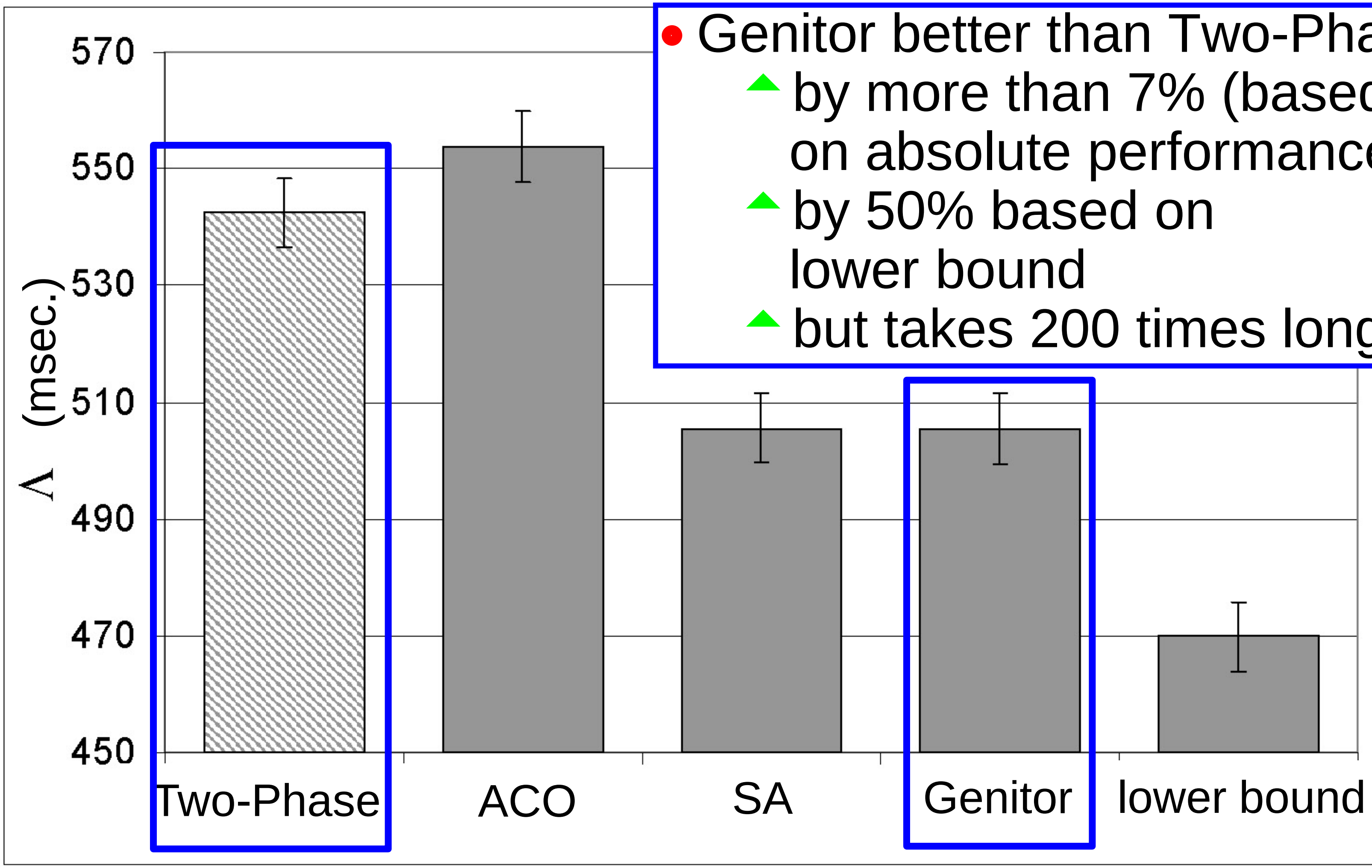
- local search applied to each offspring
  - 1. for machine with individual highest Λ
    - consider moving each task to other machines
    - if improvement, move the task that gives smallest overall system Λ
  - 2. repeat 1 until no more improvement



Recall: Procedure for Genitor

- generate initial population
- **while** stopping criterion
  - select two parent chromosomes from the 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

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

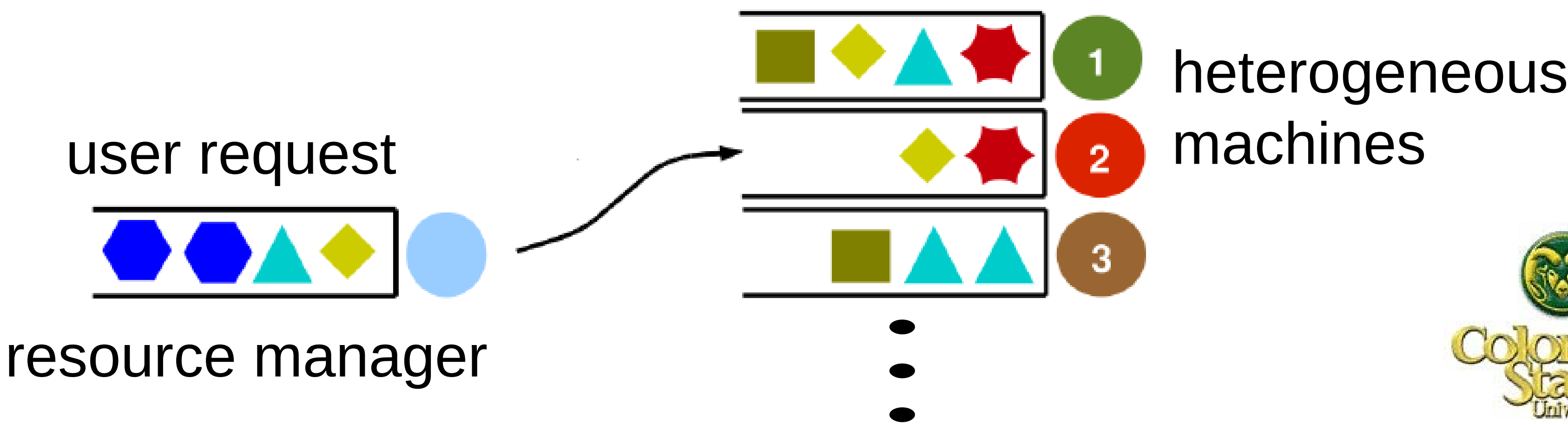
- N = 128 tasks, M = 8 machines, SRM value set to 90%
- 50 simulation trials, different PMFs for task/machine pairs
- 95% confidence intervals shown

Outline

- definition of robustness
- stochastic model and metric for robustness
- integration into static resource allocation heuristics
- use of model for a dynamic environment
- conclusions

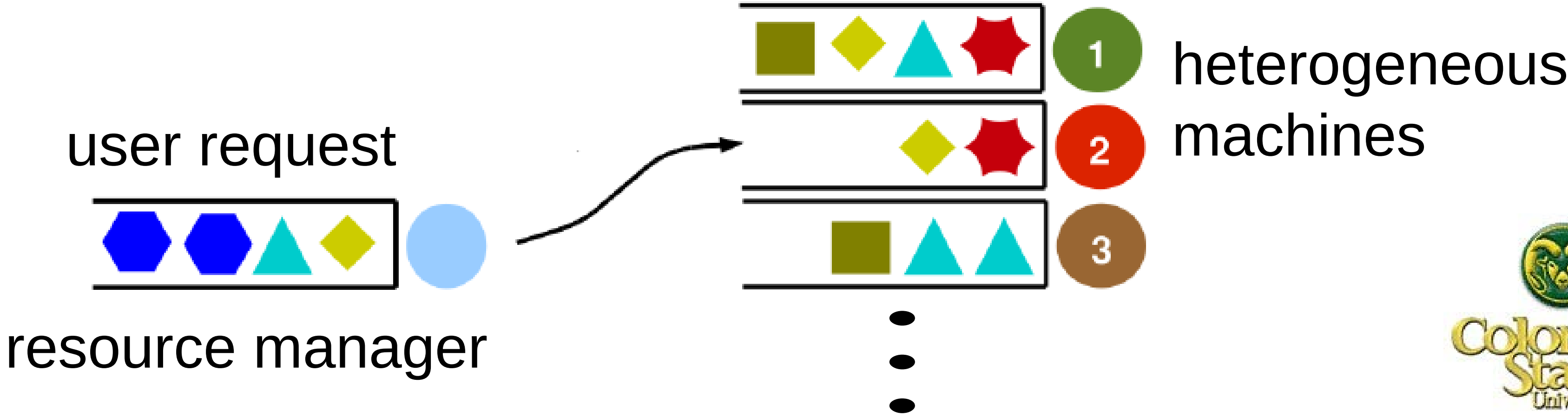
Dynamic System Model

- modeled after real-world satellite imagery processing system
- cluster of *M* heterogeneous machines
- each dynamically arriving user request has three elements
  - which existing utility application to be executed
  - archived data to be processed by that application
  - individual deadline for completing that particular request
    - agreement between service provider and customer
      - if miss deadline, complete on a “best effort” basis
- resource manager assigns requests to machines



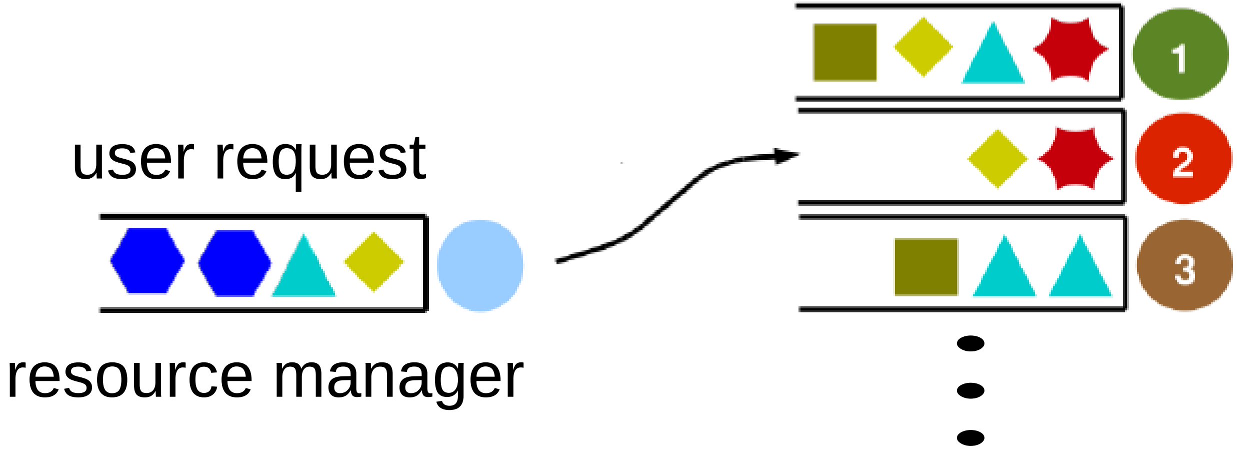
Dynamic System Performance Goal

- application execution time dependent on data size and content
- probability mass functions (PMFs) for each application's execution time on each machine, based on experiential data
- no inter-application communication
- requests cannot be re-assigned
- assume data needed for request is staged to machine while request in queue
- **goal:** complete all requests by their individual deadlines
  - late requests will be completed on "best effort" basis



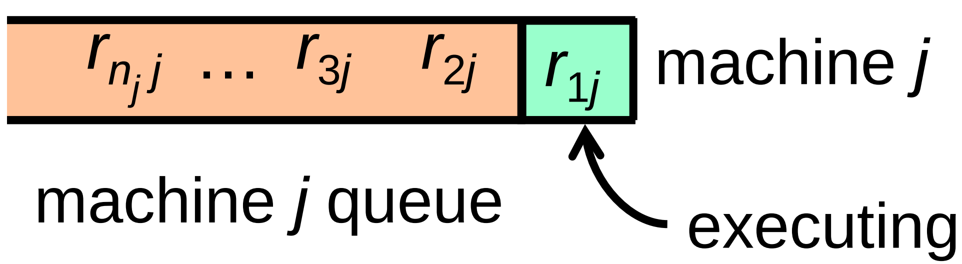
Three Robustness Questions for Dynamic System

- what behavior makes the system robust?
  - completing all requests by their individual deadlines
- what uncertainty is the system is robust against?
  - application execution times may vary substantially
- how is robustness of the system quantified?
  - probability of completing all requests by their individual deadlines

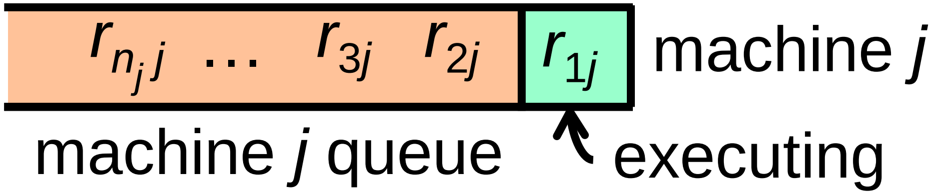


Probability of Completing All Requests by Deadlines

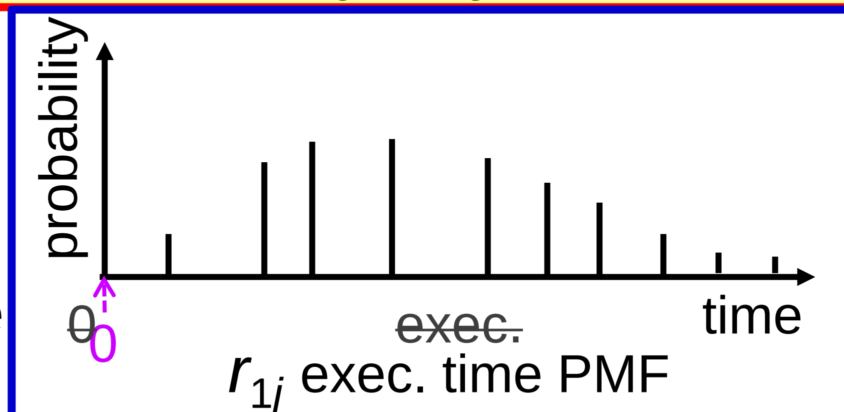
- a new mapping event is to occur at time-step $t^{(k)}$
- r_{ij} – i th request assigned to machine j at time-step $t^{(k)}$
- p(r_{ij}) – probability of completing r_{ij} by its deadline
- n_j – number of requests assigned to machine j at time-step $t^{(k)}$
- p(r_{1j} , r_{2j} , … , $r_{n_j j}$) – joint probability of completing all requests assigned to machine j by their individual deadlines

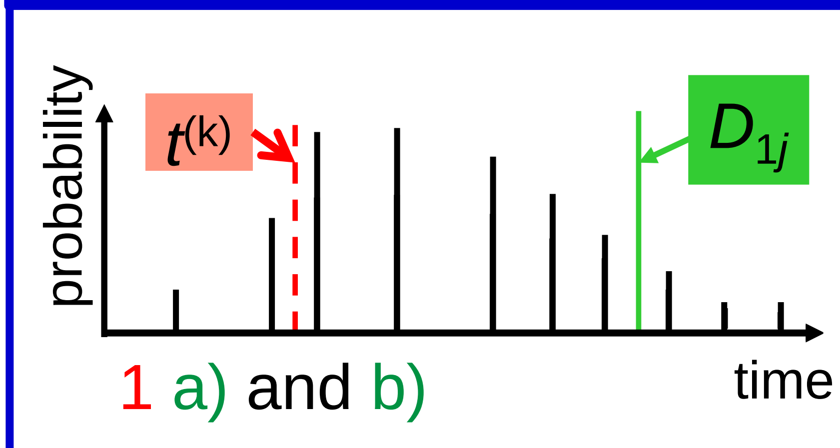


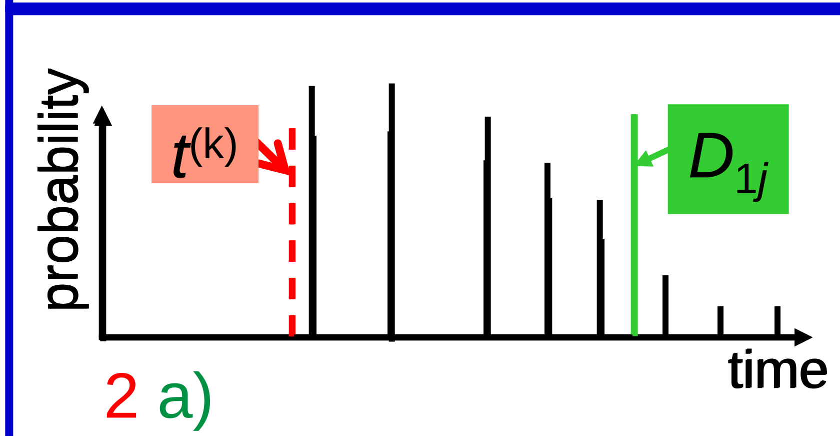
Calculating Joint Probabilities — p(r_{1j} , r_{2j})



1. find p(r_{1j}): prob. r_{1j} meets deadline
   a) $t^{(k)}$ = current time
      - drop pulses < $t^{(k)}$
      - renormalize
   b) sum pulses < deadline D_{1j}
2. find p(r_{1j}, r_{2j}) = p(r_{1j}) · p(r_{2j} | r_{1j})
   a) find PMF for r_{1j} meeting D_{1j}
      - drop pulses > deadline D_{1j}
      - renormalize
   b) convolve with execution time PMF for r_{2j}
   c) p(r_{2j} | r_{1j}) = [sum pulses < deadline D_{2j}]







Dynamic Stochastic Robustness Metric

$r_{n_j j}$ … r_{3j} r_{2j}	r_{1j}	machine j
machine j queue	executing	

- find probability to complete all requests $p(r_{1j}, r_{2j}, \ldots, r_{n_j j})$

$$p(r_{1j}, r_{2j}) = p(r_{1j}) \cdot p(r_{2j} \mid r_{1j})$$

$$p(r_{1j}, r_{2j}, r_{3j}) = p(r_{1j}, r_{2j}) \cdot p(r_{3j} \mid r_{1j}, r_{2j})$$

$$\vdots \quad = \quad \vdots$$

$$p(r_{1j}, r_{2j}, \ldots, r_{n_j j}) = p(r_{1j}, r_{2j}, \ldots, r_{n_j - 1\, j}) \cdot p(r_{n_j j} \mid r_{1j}, r_{2j}, \ldots, r_{n_j - 1\, j})$$

- $\rho^{(k)}$ – stochastic robustness metric at time-step $t^{(k)}$

$$\rho^{(k)} = \prod_{1 \le j \le M} p(\ _{1j}\ \ _{2j}\ \ \ _{n_j j})$$

- we use $\rho^{(k)}$ in dynamic resource allocation heuristics

Colorado State University

Outline

- definition of robustness
- stochastic model and metric for robustness
- integration into static resource allocation heuristics
- use of model for a dynamic environment
- conclusions

Current and Future Research

- methods to build the **initial PMFs**
- **update PMFs** using experiential data
- effective techniques for **convolving** PMFs
- incorporating stochastic robustness into static and dynamic resource allocation **heuristics** for different environments
- considering **energy** or power as a performance or constraint
- combining PMFs and probabilities when **not independent**
  - ex. **DAG** of communicating tasks
- use **relative probabilistic** information about uncertainty values
- how to combine the PMFs from **multiple uncertainties** to calculate single SRM
- how to be robust with respect to **inaccuracies** in the PMFs

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?
- devised a stochastic model for robust resource allocation
- used stochastic robustness in resource allocation heuristics
- listed areas for future research in robustness
- please see our papers listed at
  www.engr.colostate.edu/~hj/Robust_Papers.pdf
  for more information and references to other relevant research

Colorado State University