Dynamic Robust Resource Allocation in a Heterogeneous Distributed Computing System

Prof. H. J. Siegel

Abell Endowed Chair Distinguished Professor of Electrical and Computer Engineering and Professor of Computer Science

Co-Director, Center for Robust Computing

Colorado State University

Fort Collins, Colorado, USA

Outline

- introduction and system model
- robustness model and metric
- resource allocation heuristics
- simulation setup and results
- summary and next steps

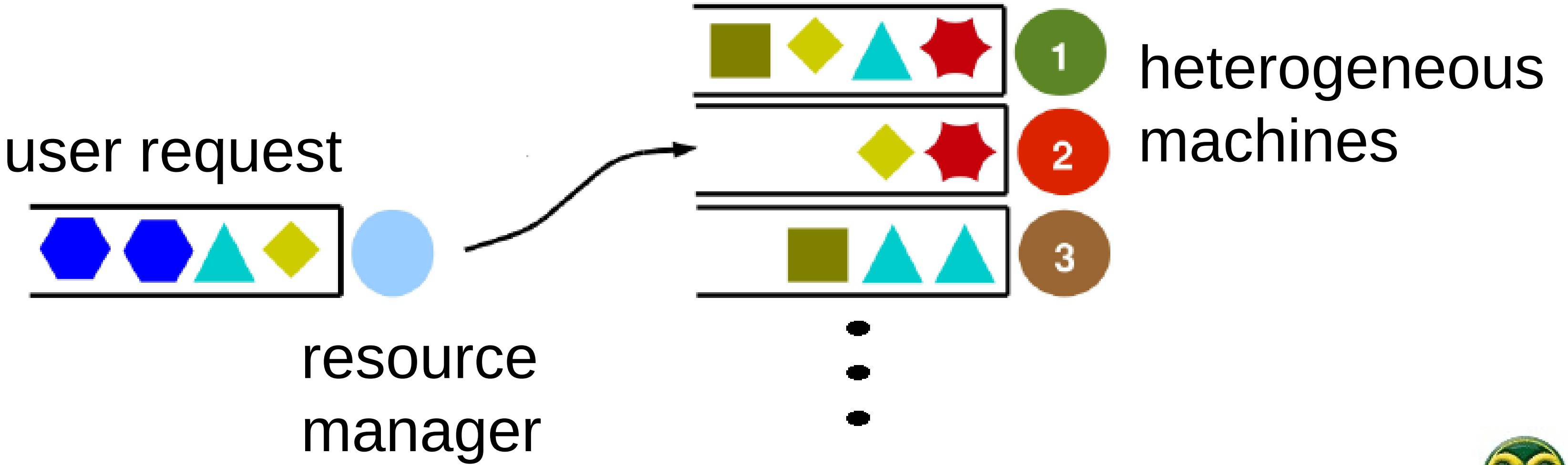
Contributions of this Research

- a mathematical model for quantifying the stochastic robustness of resource allocations in a dynamic environment

- the design of a novel resource allocation technique based on this model of robustness

Problem Statement

- modeled after real-world satellite imagery processing system
- receive user requests for image processing
- utilize cluster of *M* heterogeneous machines
  to process a dynamically arriving workload
- resource manager assigns requests to heterogeneous machines
  - requests are queued for processing

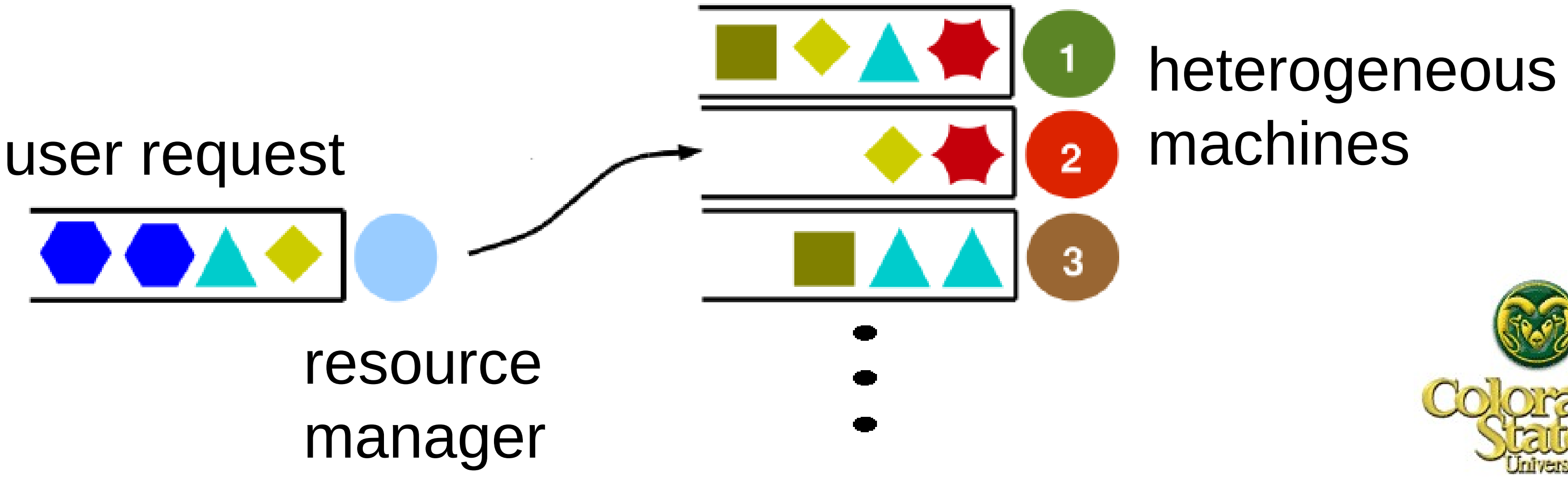


Heterogeneous Parallel Computing System

- interconnected set of different types of machines with varied computational capabilities
- workload of applications with different computational requirements
- each application may perform differently on each machine
  - furthermore: machine A can be better than machine B for application 1 but not for application 2
- **resource allocation**:
  assign requests to machines
  to optimize some performance measure
  - NP-complete (cannot find optimal in reasonable time)
  - use heuristics to find near optimal allocation

Dynamic System Model

- each dynamically arriving user request has three elements
  - which existing utility application to be executed
  - archived data to be processed by that application
  - a deadline for completing that particular request
    - agreement between service provider and customer
      - if miss deadline, complete on a “best effort” basis
- simplifying assumption that data needed for request is staged to machine while request in queue

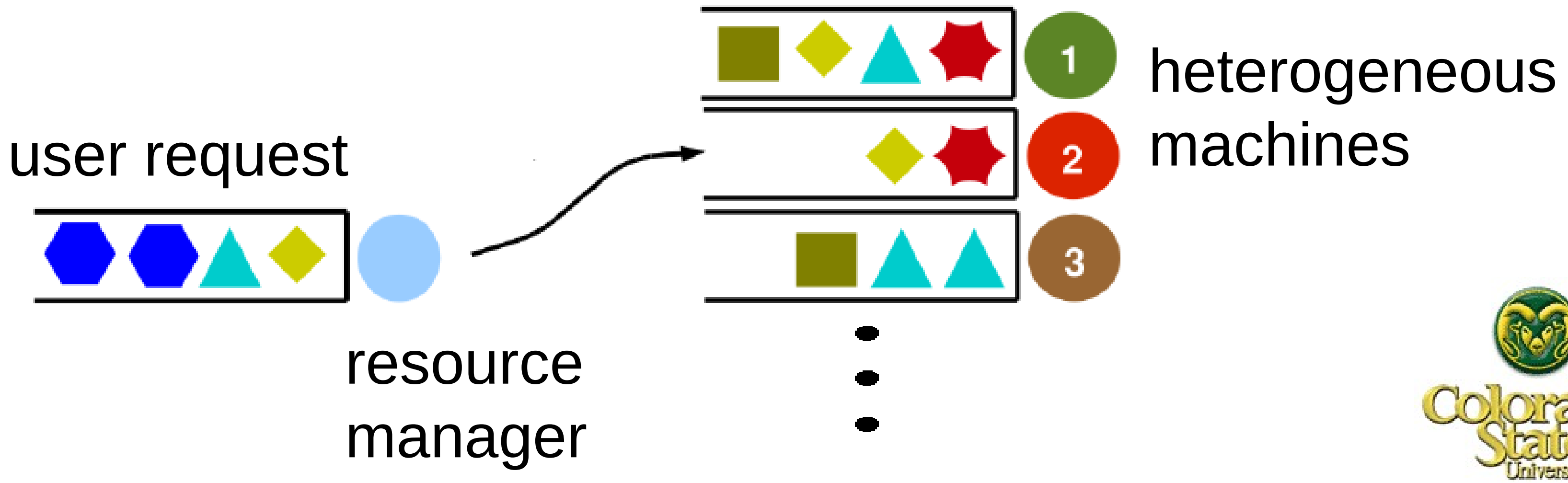


Characteristics of Applications

- applications limited to a large set of frequently run algorithms
- no inter-application communication
- application execution times may vary substantially
  - execution time dependent on data size and content, and machine assigned to application
  - modeled as "random variables"
- probability mass functions (PMFs) are provided for the execution time of each application on each machine
  - PMFs based on experiments and/or historical data
  - probability of all possible execution times for that application on that machine
  - assume accurate PMFs exist

Performance Metric

- **goal:** complete all requests by their individual deadlines
- **performance metric:**
  percent of requests that meet their individual deadlines
- dynamic immediate mode mappings considered
  - request mapped as soon as it arrives
- requests cannot be re-assigned
- queued request executed even though it cannot be completed by its individual deadline - “best effort” basis



Outline

- introduction and system model
- robustness model and metric
- resource allocation heuristics
- simulation setup and results
- summary and next steps

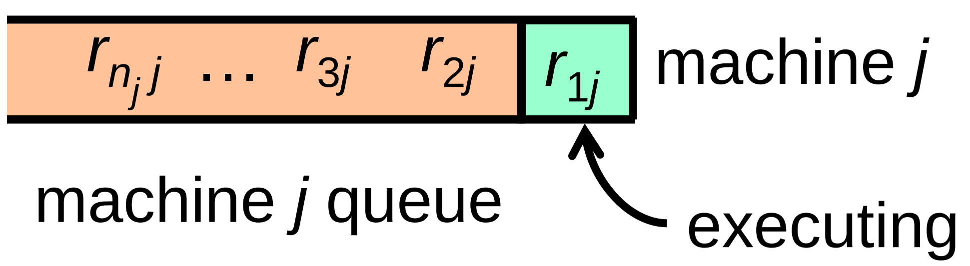
Defining Robustness for Resource Allocation

- complex computing and communication systems often operate in an unpredictable environment
  - satellite imagery processing system is just one example
- term "robustness" usually used without explicit definition
- **The Three Robustness Questions**
  1. what behavior of the system makes it robust?
     - ex. completing all requests by their individual deadlines
  2. what uncertainty is the system robust against?
     - ex. application execution times may vary substantially
  3. quantitatively, exactly how robust is the system?
     - probability of completing all requests by their individual deadlines

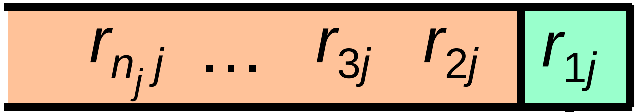
Colorado State University

Probability of Completing All Requests by Deadlines

- a new request arrives at time-step $t^{(k)}$ and needs to be assigned to a machine
- 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 $t^{(k)}$
- $p(r_{1j}, r_{2j}, \ldots, r_{n_j j})$ – joint probability of completing all requests assigned to machine j by their individual deadlines

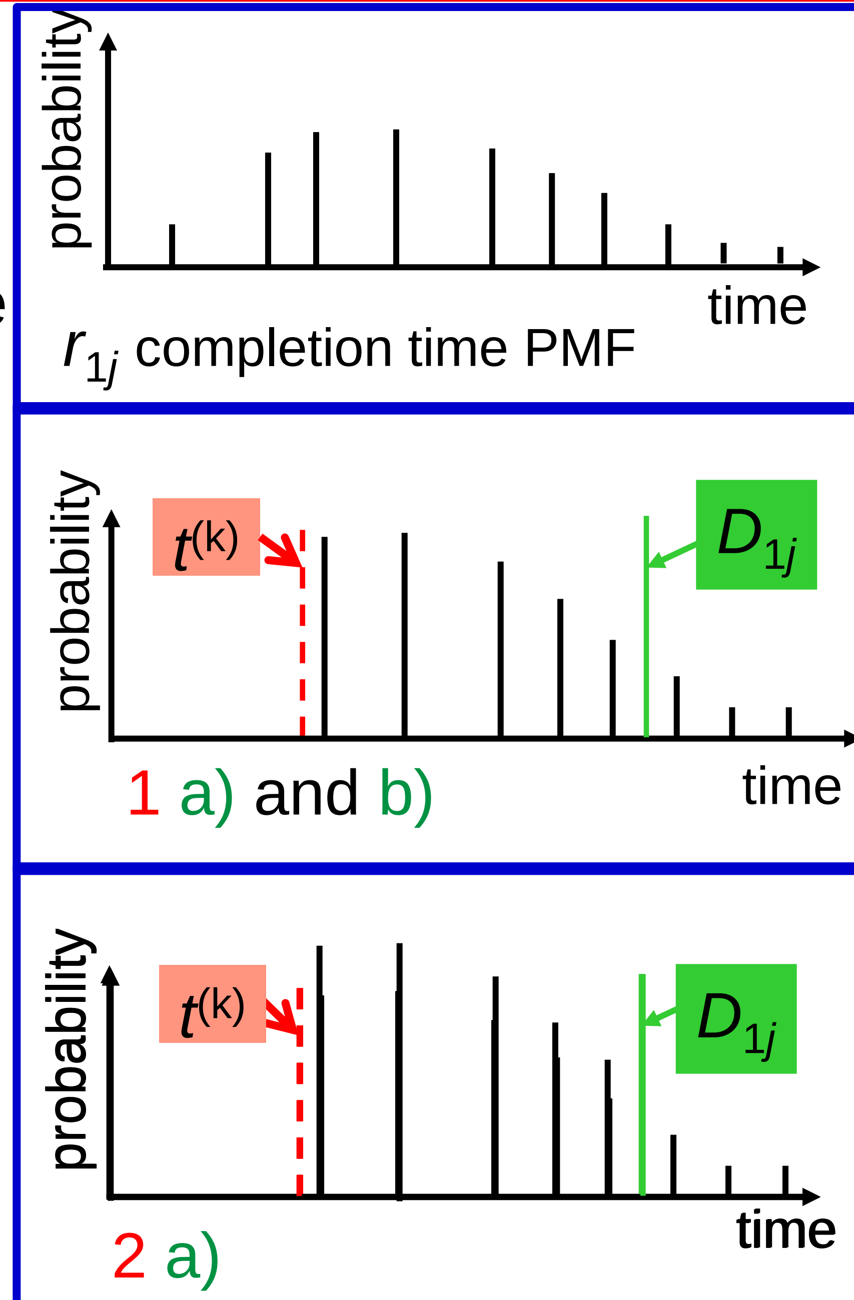


Calculating Joint Probabilities — p(r_{1j} , r_{2j})



machine j queue executing

1. find p(r_{1j}): prob. r_{1j} meets deadline
   a) drop pulses < $t^{(k)}$ (current time) and 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

- 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 $t^{(k)}$

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

Wall Clock Time Needed to Calculate $\rho^{(k)}$

- most time-consuming calculation is the convolution of the application execution time PMFs
- timed several completion time calculations on Graphics Processing Units (GPUs)
  - convolution using discrete fast Fourier transforms
    - CUFFT package from NVIDIA
  - average execution time for $\rho^{(k)}$ was 0.0029 seconds
    - using data from our experiment
    - significant reduction from general purpose CPUs
    - convolutions in real time are feasible

Outline

- introduction and system model
- robustness model and metric
- resource allocation heuristics
- simulation setup and results
- summary and next steps

Heuristics

- recall
  - **performance metric:**

    percent of requests that meet their individual deadlines
  - immediate mode heuristic
    - request assigned immediately upon its arrival
- we propose a new technique based on

  maximizing stochastic robustness
- compare with four well known resource allocation techniques
- simulation study of a heterogeneous parallel computing system

MaxRobust

- attempts to greedily maximize robustness of each request

- <u>procedure:</u>

1) for incoming request *i*
   - for each machine *j*
     - calculate $\rho^{(k)}$ ***<u>if</u>*** request *i* was added to machine *j* queue
2) assign request to machine that maximizes $\rho^{(k)}$
   - break ties using the KPB heuristic

recall: $\rho^{(k)}$ is the stochastic robustness at time-step $t^{(k)}$

Minimum Expected Completion Time (MECT)

- based on Minimum Completion Time (MCT) heuristic
- attempts to minimize the expected completion time
- because immediate mode, also implicitly attempts to maximize chance of making deadline

- procedure:

1) for incoming request *i*
   - for each machine *j*
     - calculate expected (mean) completion time ***if*** request *i* was added to machine *j* queue (use expected execution times for all requests)
2) assign request to machine that minimizes expected completion time

Minimum Expected Execution Time (MEET)

- based on Minimum Execution Time (MET) heuristic
- attempts to minimize the expected execution time of each request
- <u>procedure:</u>

1) for incoming request *i*
   - for each machine *j*
     - calculate expected (mean) execution time for request *i* on machine *j* (independent of requests already assigned to machines)
2) assign request to machine that minimizes expected execution time

K-Percent Best (KPB)

- attempts to minimize expected completion time of each request
  - uses only K% of fastest machines for a given request
    - best K% was 37.5% - 3 out of 8 machines (determined empirically)
- because immediate mode, also implicitly attempts to maximize chance of making deadline
- <u>procedure:</u>

1) for incoming request *i*
   - identify the K best set of machines ($Best_k$)
   - for each machine $j \in Best_k$
     - calculate expected completion time ***<u>if</u>*** request *i* was added to machine *j* queue (use expected execution times for all requests)

2) assign request to machine that minimizes expected completion time

Shortest Queue (SQ)

- assigns requests to machines with the smallest number of requests in the queue

procedure:

1) assign *i* to the machine with the smallest number of pending requests in its input queue
    - ties are broken arbitrarily

Outline

- introduction and system model
- robustness model and metric
- resource allocation heuristics
- simulation setup and results
- summary and next steps

Simulation Setup — Machine Description

- system of eight heterogeneous machines
- assumed 12 different application types
  - SPECInt benchmark application results used to simulate execution time PMFs
- each simulation trial
  - 2,000 dynamically arriving requests
  - requests arrived over period of 20,000 time-steps
  - modeled arrivals as a Poisson process
- deadline for each request = arrival time + average over all machines of expected execution time (tight)

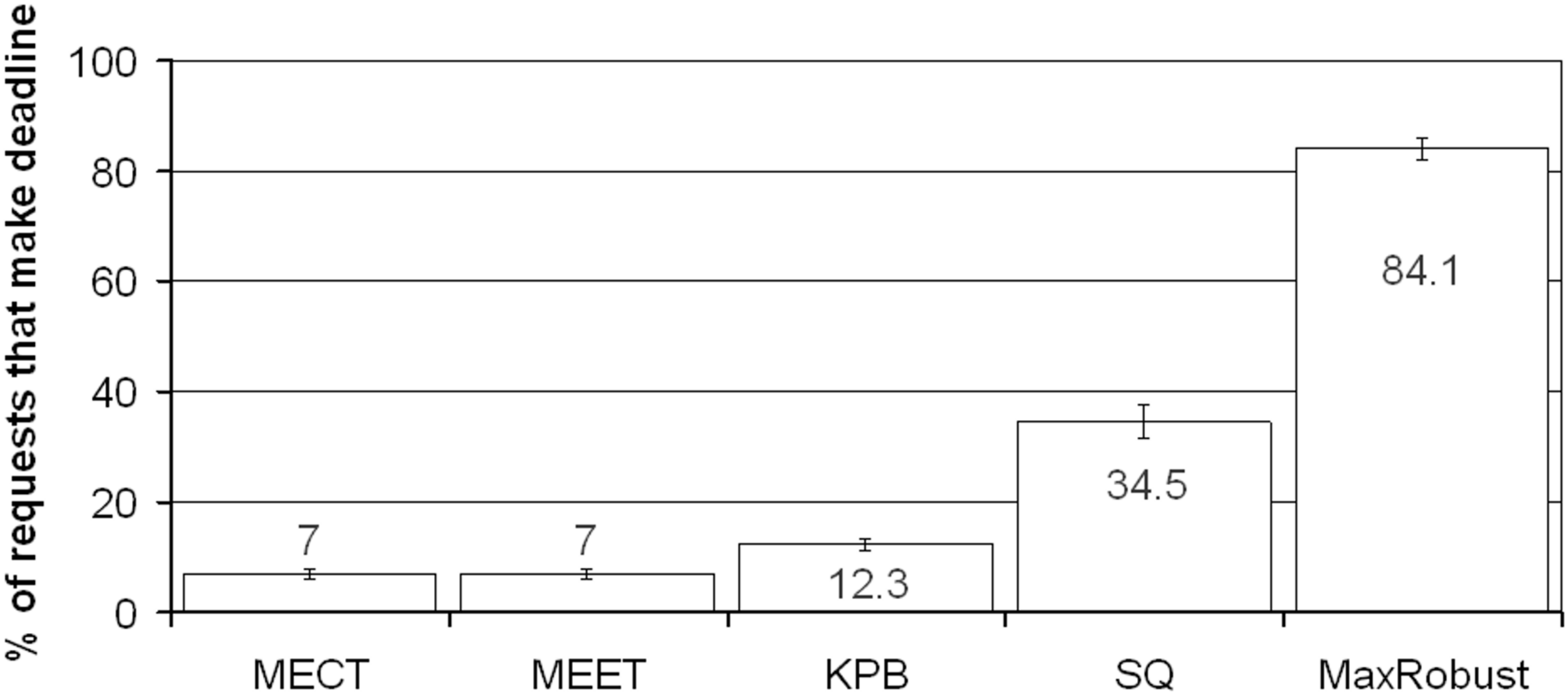
note: SPECint is the integer performance testing component of the Standard Performance Evaluation Corporation (SPEC) test suite

Colorado State University

Simulation Setup — Simulation Trials

- reported results for 100 different simulation trials
  - each request randomly assigned application type (1 through 12)
  - simulated execution times sampled from application execution time PMF
    - actual execution times in the simulation
    - used to determine if application met deadline

Comparison of Heuristic Results



- MECT – Minimum Expected Completion Time
- MEET – Minimum Expected Execution Time
- KPB – K-Percent Best
- SQ – Shortest Queue

Colorado State University

Discussion of Results — Arrival of First Requests

- for all heuristics, requests were likely to meet their deadline at the beginning of the simulation
  - arrival of first 50 requests
  - initially machines are more likely to complete requests assigned to them
    - machines start in idle state
    - during start-up machines are undersubscribed

Discussion of Results — MaxRobust

- MaxRobust performed significantly better than other heuristics
  - only heuristic to use stochastic information
  - only heuristic to use explicitly information about deadlines

Discussion of Results — MEET

- Minimum Expected Execution Time (MEET)
- MEET performed poorly
  - ignored stochastic information
  - MEET underutilized poor performing machines

Discussion of Results — MECT and KPB

- Minimum Expected Completion Time (MECT)
- MECT performed poorly
  - ignored stochastic information
  - if request takes longer than expected,

    then other requests in the queue may miss their deadline

    even if they do not take longer than expected times
- K-Percent Best (KPB)
- KPB better than MECT because used subset of MET machines
  - but still had MECT problems

Discussion of Results — SQ

- Shortest Queue (SQ)
- SQ performed significantly better than KPB, MECT, and MEET
  - not as good as MaxRobust
  - selecting machine with shortest queue reduces impact of some requests having a longer than expected execution time
    - minimizes number of preceding requests in queue on average

Outline

- introduction and system model
- robustness model and metric
- resource allocation heuristics
- simulation setup and results
- summary and next steps

Summary

- designed a mathematical model for quantifying the stochastic robustness of resource allocations in a dynamic environment
- designed and evaluated MaxRobust heuristic
  - based on stochastic robustness
- MaxRobust performs significantly better than SQ, MECT, MEET, and KPB
  - MECT and KPB are adapted from heuristics that have been shown to perform well in other problems
  - MaxRobust heuristic has shown promise in our experiments
  - results shows importance of stochastic robustness in dynamic environments

Next Steps

- methods to collect data to build the initial PMFs
- methods to update PMFs using experiential data
- fast and effective techniques for convolving PMFs
- consider batch-mode heuristics in this environment
- consider how to manage situations when joint probability is 0
- evaluate importance of accurate PMFs

Colorado State University

Reference

- “Stochastic-Based Dynamic Resource Allocation in a Heterogeneous Computing System”
- by Smith, Chong, Maciejewski, and Siegel
- *38th International Conference on Parallel Processing (ICPP 2009)*
- pp. 188-195
- Sep., 2009