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

THE MARKETING DEPARTMENT HAS ASKED US TO MAKE OUR PRODUCTS MORE ROBUST.

NONE OF US KNOWS WHAT THAT MEANS.
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

**Examples:**

- **Intel Phi Coprocessor**
- **Cray XC-30 Blades**
- **HP BladeSystem C7000**
- **Nvidia Tesla GPU**
- **Hitachi Blade Server 500**
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
    - $5^{30}$ 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
sensors produce periodic data sets, each with multiple data files

\( N \) independent tasks process each data set within \( \Lambda \) time units

\( N \) tasks statically mapped to \( M \) heterogeneous machines, \( N > M \)

similar: satellite data maps, security surveillance

**Ex.: Radar Data Processing for Weather Forecasting**
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 ($\Lambda$) between data sets
  - **constraint**: a user-specified probability of 90% that all tasks will complete in $\Lambda$ 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 ($\Lambda$) between data sets
  - **constraint**: a user-specified probability of 90% that all tasks will complete in $\Lambda$ 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 $\Lambda$ 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 $\Lambda$ time units for every data set
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
  - \[ \text{probability} = \frac{\text{frequency}}{\text{total \# samples}} \]
- example: probability of value from 10 to 19 = \( \frac{6}{200} = 3\% \)
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
• “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

\[ p(\tau_A = k) = \sum_{\tau_1 + \tau_2 = k} (p(\tau_1) \cdot p(\tau_2)) \]
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

\[ p(\tau_A = k) = \sum_{\tau_1 + \tau_2 = k} (p(\tau_1) \cdot p(\tau_2)) \]
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

\[ p(\tau_A = k) = \sum_{\tau_1 + \tau_2 = k} (p(\tau_1) \cdot p(\tau_2)) \]

1 + 3 = 4
• 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 Completion Time of Machine

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

\[ p(\tau_A = k) = \sum_{\tau_1 + \tau_2 = k} (p(\tau_1) \cdot p(\tau_2)) \]
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 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

- $\Lambda$: deadline for completing all tasks
- machine $j$ stochastic robustness $\text{Prob}[S_j \leq \Lambda]$
- **Stochastic Robustness Metric (SRM)**
  \[
  \prod_{j=1}^{M} \text{Prob}[S_j \leq \Lambda]
  \]
- **goal of heuristics**
  \[\uparrow \text{minimize } \Lambda \text{ for a given SRM value}\]
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 $\Lambda$ 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 $\Lambda$

  - **phase 2:** among these task/machine pairs
    - find task/machine pair with minimum $\Lambda$
    - map this task to its associated machine
Heuristic: Genitor Genetic Algorithm

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

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>...</td>
</tr>
</tbody>
</table>

• population size of 200 (decided empirically)

• initial population generation
  \( \uparrow \) one chromosome: solution from the Two-Phase Greedy heuristic (“seed”)
  \( \uparrow \) other 199: simple greedy heuristic

• population in ascending order based on minimum $\Lambda$ 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 \( \Lambda \) 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

Parents:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Offspring:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Genitor: Crossover

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Genitor: Mutation

- 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

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \ldots \\
2 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 2 & \ldots \\
\end{array}
\]
Genitor: Local Search

- Local search applied to each offspring
  - 1. For machine with individual highest $\Lambda$
    - Consider moving each task to other machines
    - If improvement, move the task that gives smallest overall system $\Lambda$
  - 2. Repeat 1 until no more improvement
Simulations: Performance of Static Heuristics

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

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

![Diagram of mapping event]

- Tasks that arrived since last mapping event:
  - t_{10}, t_{11}, t_{12}, t_{13}, t_{14}

- Virtual queue in the scheduler:
  - t_7, t_2, t_5

- Tasks in machine queues:
  - t_3, t_1, m_1
  - t_4, m_2
  - t_8, m_3
  - t_6, m_4

- Mappable tasks set: tasks that arrived since last mapping 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

\[ \sum \text{all tasks} \]

probability that task \( i \) completes by its deadline

expected \# tasks that will complete by their individual deadlines
Probability Completing Executing Task by Deadline

- Machine $j$ queue
- Executing
- New mapping event time $k$
- $\rho(t_{1j})$: probability of $t_{1j}$ completing by its deadline
  a) Time $k = \text{current time}$
     - Drop pulses $< k$
     - Renormalize
  b) Sum pulses $< \text{deadline } D_{1j}$
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

![Graphs showing PMF for $t_{ij}$ and $t_{(i-1)j}$](image-url)
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

**Graphs:**
- Execution PMF for $t_{ij}$
- Completion PMF for $t_{(i-1)j}$ (if task 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 $\rho(t_{ij})$

![Completion PMF Graphs](image)
Stochastic Robustness for Dynamic Heuristics

\[ \rho(t_{ij}) \]

probability that task \( t_{ij} \) completes before its deadline

\[ \sum_i \rho(t_{ij}) \]

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

\[ \rho_j(k) \]

\[ \sum_j \rho_j(k) \]

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

**stochastic dynamic robustness:** \( \rho(k) \)

the expected number of tasks that will meet their deadlines measured at time \( k \)
Heuristic: Maximum On-time Completions (MOC)

- during a mapping event at time $k$

  for each task $t_a$ in mappable tasks
    find machine that maximizes overall $\rho(k)$
    add $t_a$ to the list of candidate tasks for that machine
    first phase

    for each machine $j$ with a queue that is not full
      get candidate task for that machine with maximum $\rho(k)$
      assign that task and remove from mappable tasks
      second phase
**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

- **Heuristic: Min Completion - Max Urgency (MMU)**
  - **phase 2:** map task in task/machine pair that maximizes urgency $= 1 / (\text{task deadline} - \text{expected completion time})$

- **Heuristic: Min Completion - Soonest Deadline (MSD)**
  - **phase 2:** map task in task/machine pair with the soonest deadline
Results: Varied Deadline Weight Parameter ($w$)

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

- 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

% tasks completed on time

0 10 20 30 40 50 60 70 80 90

0.1 0.5 1 2

deadline weight parameter $w$
Results: Varied Number of Tasks in Workload

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

- **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

![Bar chart showing the percentage of tasks completed on time for different methods with varying number of tasks. The methods compared are MOC, MM, MSD, and MMU. The chart shows that MOC performs best, followed by MM, with MMU and MSD performing worse.](image-url)
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)

- histogram
- 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

![Diagram showing probability distribution]
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

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

![Graphs showing completion PMFs for tasks and deadlines.]
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

\[ \sum_{ij} \rho(t_{ij}) \]
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
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
definition and stochastic model of robustness


use in static resource allocation heuristics


use in dynamic resource allocation heuristics


sponsors of this research

- NSF
- Department of Defense
- Oak Ridge National Laboratory
- Colorado State University