Parallel Computing 35 (2009) 389–400

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Robust resource allocation in a cluster based imaging system ☆

Jay Smith [a,b,*], Vladimir Shestak [b,d], Howard Jay Siegel [b,c], Suzy Price [d], Larry Teklits [d], Prasanna Sugavanam [b]

[a] *DigitalGlobe, Longmont, CO 80503, USA*
[b] *Dept. of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523-1373, USA*
[c] *Dept. of Computer Science, Colorado State University, Fort Collins, CO 80523-1373, USA*
[d] *InfoPrint Solutions Company, Boulder, CO 80301, USA*

A R T I C L E I N F O

Article history:
Received 1 October 2008
Accepted 31 March 2009
Available online 7 April 2009

Keywords:
Heterogeneous computing
Dynamic resource allocation
Robustness

A B S T R A C T

Recently there has been an increased demand for imaging systems in support of high-speed digital printing. The required increase in performance in support of such systems can be accomplished through an effective parallel execution of image processing applications in a distributed cluster computing environment. The output of the system must be presented to a raster based display at regular intervals, effectively establishing a hard deadline for the production of each image. Failure to complete a rasterization task before its deadline will result in an interruption of service that is unacceptable. The goal of this research was to derive a metric for measuring robustness in this environment and to design a resource allocation heuristic capable of completing each rasterization task before its assigned deadline, thus, preventing any service interruptions. We present a mathematical model of such a cluster based raster imaging system, derive a robustness metric for evaluating heuristics in this environment, and demonstrate using the metric to make resource allocation decisions. The heuristics are evaluated within a simulation of the studied raster imaging system. We clearly demonstrate the effectiveness of the heuristics by comparing their results with the results of a resource allocation heuristic commonly used in this type of system.

1. Introduction

Recently there has been an increased demand for imaging systems in support of high-speed color digital printing. Increases in print speeds and resolution have necessitated a significant increase in the performance of imaging systems in support of digital printing systems. This required increase in performance can be achieved through an effective parallel execution of image processing applications in a distributed computing environment. In this paper, we present a mathematical model of a distributed raster imaging system, where the output of the system must be presented to a raster based display at fixed regular time intervals, effectively establishing a hard deadline for the completion of each output image. This mathematical model is used as the basis for the design of a resource allocation heuristic applicable to this distributed computing environment. We extend the use of our model by deriving a robustness metric appropriate to this environment. This robustness metric is used within our presented resource allocation heuristic as an alternate optimization criterion for the heuristic.

☆ The robustness portions of this research were supported by the NSF under Grant CNS-0615170 and by the Colorado State University George T. Abell Endowment.

* Corresponding author. Address: DigitalGlobe, Longmont, CO 80503, USA.

E-mail addresses: jtsmith@digitalglobe.com (J. Smith), vladimir.shestak@infoprint.com (V. Shestak), hj@colostate.edu (H.J. Siegel), suzy.price@infoprint.com (S. Price), larry.teklits@infoprint.com (L. Teklits).

0167-8191/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2009.03.001

In this system, an input stream of data, described using a high level language known as a page description language (*pdl*), e.g., postscript or the portable document format (pdf), arrives at an imaging system for rasterization [8]. Rasterization of pdl images converts the images from a pdl description to a *bitmap*. Requests for rasterization of pdl images are processed by a dedicated cluster of workstations, where individual pdl image requests, referred to as *sheetsides*, are distributed to the heterogeneous cluster by a centralized image dispatcher. The collection of sheetside requests together describe an image stream that is displayed on a raster based device, e.g., a printer or computer monitor. The frequency of requests and the magnitude of the data required to describe each request pose a considerable challenge for even modern workstations. Input streams in this environment routinely consist of over 100,000 images, where each image typically requires 10–100 megabytes of storage and successive image deadlines are on the order of a tenth of a second apart.

For the studied environment, the images that comprise the input datastream are required to be displayed *in order* on the output device. That is, each pdl image has a unique number assigning its place in the overall stream of images and will be requested by the raster display in that order. In addition, the system has a finite amount of storage capacity (distributed evenly across the cluster of workstations) in which to store rasterized images. The bitmaps to be displayed are retrieved at a regular interval directly from the workstation output buffers by the display device. Bitmaps are displayed for a fixed time interval, thus, the display time of the first bitmap establishes a hard deadline for each subsequent bitmap. Missing a deadline for a required bitmap results in an interruption of service that is unacceptable.

The studied rasterization system has some additional special requirements that complicate the task of assigning the stream of incoming pdl images to available workstations. The computation required to convert a pdl image to a bitmap depends on the content of the pdl file. The system only has an estimate of the time required to rasterize each incoming pdl image on each type of processor and this estimate may differ substantially from the actual time required for rasterization. Many of the system design decisions are motivated by an attempt to mitigate the impact of this uncertainty.

Second, the overall system has finite input and output storage capacity, thus, there is a limit on the number of pdl images that can be buffered in the system, both as input pdl images and as output bitmaps. Finally, pdl images continue to arrive for rasterization while others are being rasterized, i.e., the resource allocation must be produced dynamically [13]. The general problem of assigning tasks to workstations in a dynamic environment has been shown to be NP-complete (e.g., [4,5,9]). Consequently, the design of heuristics for dynamic resource allocation is an active area of research [3,6,13,15,20,23].

The concept of robustness of a resource allocation was introduced in [1,18], and later efforts applied the concept to resource management [15,18,20,21,2,17]. The mathematical model presented in this work builds upon principles addressed in our earlier work on robustness. In analyzing this image processing system, we identified that rasterization times are a source of uncertainty in the system and that the arrival ordering of sheetsides is not known *a priori*. This uncertainty can impact the system by causing sheetsides to miss their deadline, resulting in an interruption of service that is unacceptable in this system. Clearly, the area of robust operation within this system exists where bitmaps are always available in advance of their deadlines.

The primary contributions of this work are: (1) a mathematical model of a distributed raster image processing system, (2) the derivation of a robustness metric for a dynamic distributed computing system with hard deadlines for task completions, and (3) the design of the resource allocation heuristics suitable for this type of system. We clearly demonstrate the superiority of our heuristic technique (using two different optimization criteria) over a technique commonly used in this type of environment.

The details of the system model that motivated this research are given in the next Section. This system model is used to design a mathematical model of rasterization completion times presented in Section 3. Section 4 describes a new resource allocation heuristic that incorporates this mathematical model for rasterization completion times. We present a discussion of the performance objective for this initial heuristic in Section 5. This performance metric motivated the derivation of the robustness metric in Section 6. The details of the simulation setup are described in Section 7 and the results of the heuristics are presented in Section 8. A sampling of related work is in Section 9 and Section 10 concludes the paper. A glossary of notation and acronyms used in the paper are tabulated in the Appendix.

2. System model

Fig. 1 is a conceptual drawing of the system that motivated this research. In this environment, two display devices combine to provide a high-speed digital continuous-form, color duplex (i.e., two-sided) printer. Paper is physically moved through each press successively at high-speed and cannot be immediately stopped. Consequently, if a bitmap is not readily available in the display device when it is needed, then a service interruption occurs because the printer must be stopped to accommodate the delay and the advanced paper removed from the output of the device.

The imaging system is composed of a collection of workstations dedicated to image rasterization, controlled by a separate master workstation called the *head node*. Individual sheetsides are transferred to the head node, where they are dispatched by the centralized image dispatcher to one of the dedicated workstations for rasterization. Each sheetside completely describes an entire display image suitable for the output device, and is expressed in a logical page description language that must be transformed into a bitmap suitable for the display device.

Input sheetsides are queued for rasterization in the Head Node Input Queue (HNIQ). The centralized image dispatcher assigns sheetsides from the HNIQ to workstations for rasterization. After assignment, the head node places the sheetside

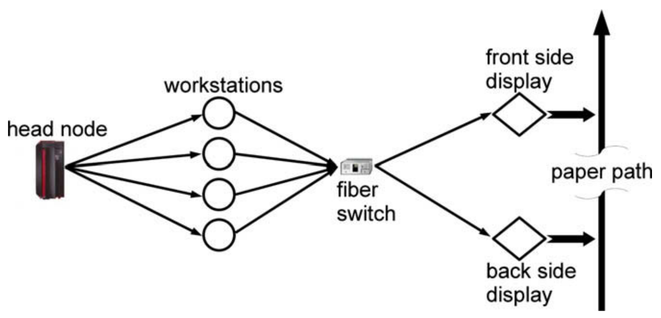

Fig. 1. A conceptual model of a high performance, cluster based imaging system.

in a queue for a transmitter that will then transmit the sheetside to its destination. The size of the transmitter queue, denoted $|TQ|$, is limited to only two sheetsides and the transmitter may only transfer one sheetside at a time. Further, once a sheetside has been placed into the transmitter queue, the destination workstation for the sheetside can no longer be modified. Each workstation has a finite capacity input buffer for storing sheetsides prior to their rasterization. Therefore, before the incoming sheetside can be placed in the transmit queue, the head node must ensure that sufficient capacity exists in the input buffer of the receiving workstation to acquire the file. If there is sufficient input buffer capacity, then the sheetside may be queued up for transmission.

It is assumed that M heterogeneous dedicated workstations are available to convert pdl sheetsides to bitmaps. Each workstation is interconnected to the head node via a 1 Gbit ethernet network and interconnected to the two output display devices using a 4 Gbit fiber channel. The memory of each workstation is divided into two blocks, where one block is used to store sheetside pdl files (the input to rasterization) and the other block is used to store output bitmaps. The sheetsides in the input block are accessed in a FIFO fashion.

When a workstation completes the rasterization of a sheetside, notification of the completion is sent to the head node, and the appropriate display device. When the display device is ready to display the bitmap, it retrieves the completed bitmap directly from the output buffer of the workstation where the rasterization was performed. Each display device has an input buffer with sufficient capacity to store two bitmaps, i.e., the bitmap currently being displayed and the next bitmap to be displayed. In this system, all bitmap files are assumed to be the same size, and the time required to display each bitmap is assumed constant.

3. Model of rasterization completion time

The mathematical model of this system defines a method for calculating the deadline for a given sheetside, and a method for determining the estimated rasterization completion time for a sheetside on a given workstation in the system at a specific point in time. The completion of every sheetside is subject to a hard deadline. That is, to prevent a service interruption, each sheetside must complete rasterization, and be available for consumption in the input buffer of the appropriate display device by its given deadline. To calculate the deadline for a sheetside, let t_0 be the absolute wall-clock start time for both display devices. Starting at t_0, each display will require a new bitmap every $t_{display}$ seconds, where $t_{display}$ is the time required to display a bitmap on the device. Sheetsides are numbered starting with 1. Incoming sheetsides are divided between the two displays such that the odd numbered sheetsides go to display 1 and the even numbered sheetsides go to display 2 (see Fig. 1). For the kth actual sheetside of the job, denoted S_k, the bitmap must be available for printing at time $t_0 + t_{display}\left(\frac{S_k-1}{2}\right)$, if k is odd, and at time $t_0 + t_{display}\left(\frac{S_k}{2}\right)$, if k is even (note: the division by 2 is because two sides are printed simultaneously). Let t_{tran}^{bitmap} be the bitmap transfer time from any workstation to either display device. Then, the *deadline* for completing S_k, denoted $t_d[S_k]$, is the latest wall-clock time for a workstation to produce the bitmap for S_k

$$t_d[S_k] = \begin{cases} t_0 + t_{display}\left(\frac{S_k-1}{2}\right) - t_{tran}^{bitmap} & \text{if } S_k \text{ is odd;} \\ t_0 + t_{display}\left(\frac{S_k}{2}\right) - t_{tran}^{bitmap} & \text{if } S_k \text{ is even.} \end{cases} \quad (1)$$

The value of $t_d[S_k]$ will be used later to determine the availability of output buffer space on a workstation. For this purpose, the deadline equation needs to be expressed in terms of the ordering of sheetsides on a given workstation. Let BQ_i^j be the ith sheetside to have entered the input queue of workstation j for a given job. We define the following operator $\text{num}(BQ_i^j)$ that evaluates to the actual sheetside number of sheetside BQ_i^j, i.e., $S_k = \text{num}(BQ_i^j)$. Then S_k in Eq. (1) can be replaced by $\text{num}(BQ_i^j)$. Note that S_k and BQ_i^j represent the same physical sheetside and by using the num operator these notations can be used interchangeably.

The estimated rasterization *completion* time for a sheetside is composed of the earliest possible time that rasterization can begin and the estimated rasterization time. Let $t_{start}\left[BQ_i^j\right]$ be the earliest possible rasterization start time for sheetside BQ_i^j, let

$ERT\left[BQ_i^j\right]$ be the estimated rasterization time for sheetside BQ_i^j, and let $t_{comp}^j\left[BQ_i^j\right]$ be the estimated rasterization completion time for a given sheetside BQ_i^j on workstation j. Then $t_{comp}^j\left[BQ_i^j\right]$ can be calculated as:

$$t_{comp}^j\left[BQ_i^j\right] = t_{start}\left[BQ_i^j\right] + ERT\left[BQ_i^j\right]. \tag{2}$$

The estimated rasterization time, $ERT\left[BQ_i^j\right]$, for each sheetside is assumed known based on empirical data. Recall that the workstations in the cluster are heterogeneous, thus, the value of $ERT\left[BQ_i^j\right]$ may differ for each workstation in the cluster. There are many well-known techniques for gathering these execution time estimates from empirical data [7,11,12,19,24]. The start time for rasterization depends on several factors: when the sheetside was transferred to the workstation, when the previous sheetside assigned to the workstation completed, and the availability of the output buffer of the workstation. The rasterization for a sheetside starts only if the output buffer has enough capacity to accommodate the resultant bitmap. During the rasterization process the amount of memory required to store a bitmap remains reserved in the output buffer. The memory held by a sheetside in the input buffer is released when rasterization for that sheetside is completed.

To determine when a sheetside was (or will be) transferred to a workstation, we have to consider all other sheetsides in the HNIQ that are ahead of it. Let S_k be the kth sheetside to enter the HNIQ for a given job, where S_{k-1} is the sheetside ahead of S_k in the HNIQ. To evaluate the estimated departure time for S_k to workstation j, the input buffer capacity of workstation j must be determined. Space in the input buffer is limited by two factors: the maximum number of sheetsides (Q) allowed in the input buffer and the size in bytes of the input buffer.

Recall that the estimated rasterization times are known to be only estimates of the actual rasterization times. The number of sheetsides that are allowed to queue up on any given workstation is limited to a relatively small, fixed number of pending sheetsides, to attempt to mitigate the impact of delays caused by under-estimating sheetside rasterization times. If the size of the pdl file describing sheetside S_k is less than or equal to the available input buffer capacity of workstation j, then, assuming there are fewer than Q sheetsides in the input buffer of workstation j, S_k can be immediately sent to j following the transmission of S_{k-1} out of the head node. Otherwise, S_k will be delayed at the head node (blocking S_z, $z > k$) for the amount of time required for a certain number of sheetsides previously assigned to workstation j to be rasterized, thus, creating buffer capacity sufficient to accommodate the pdl file of sheetside S_k or for the number of pending sheetsides on workstation j to be less than Q.

To calculate the available input buffer capacity at workstation j, let $\mathscr{K}$ be the sequence of sheetsides that are in the input buffer of workstation j when the head node transmitter is ready to send sheetside S_k. Note that this will include any sheetside currently being rasterized by workstation j. Let the operator size(S_i) give the size in bytes of the pdl file for sheetside S_i, and let CAP_{in}^j describe the total input buffer capacity of workstation j. Then, the available capacity of the input buffer for workstation j, denoted AC_{in}^j, is:

$$AC_{in}^j = CAP_{in}^j - \sum_{\forall S_i \in \mathscr{K}} \mathrm{size}(S_i). \tag{3}$$

Define $t_{dept}^x[S_{k-1}]$ as the departure time of S_{k-1} from HNIQ to workstation x. Let $t_{tran}^{sdf}[S_{k-1}]$ be the time required to transfer the sheetside description file describing S_{k-1}, from HNIQ to workstation x. If $\mathrm{size}(S_k) \leqslant AC_{in}^j$ and $|\mathscr{K}| < Q$, S_k can depart at time:

$$t_{dept}^j[S_k] = t_{dept}^x[S_{k-1}] + t_{tran}^{sdf}[S_{k-1}]. \tag{4}$$

Otherwise, sheetside S_k cannot be transmitted until a sufficient number of sheetsides have been processed from the input buffer of workstation j, to ensure that these two conditions hold. If after processing some sheetside $S_m \in \mathscr{K}$ these conditions hold, then $t_{dept}^j[S_k] = t_{comp}^j[S_m]$. If $k = 1$, i.e., S_k is the first sheetside to be dispatched by the centralized image dispatcher, then S_k can depart immediately. Fig. 2 presents concise pseudo-code for determining the earliest estimated departure time for a given sheetside.

Prior to rasterization, accurately determining when rasterization can begin for some sheetside BQ_i^j, where $S_k = \mathrm{num}(BQ_i^j)$, must also account for possible delays incurred due to the limited capacity of the output buffer on each workstation. Consider workstation j with output buffer capacity CAP_{out}^j. Because bitmaps are all assumed to be the same size, i.e., require the same number of bytes to store, the number of bitmaps that could be stored in the output buffer of any workstation is constant and known in advance. Assume that N bitmaps can be placed in the output buffer of a given workstation. Define the delay to begin processing sheetside BQ_i^j, denoted $\Delta_{out}\left[BQ_i^j\right]$, as the time that BQ_i^j must wait after arriving at the head of the input queue of workstation j until there is sufficient capacity in the output buffer of the workstation to store the output bitmap. To quantitatively determine $\Delta_{out}\left[BQ_i^j\right]$, there are three cases to consider. First, if fewer than N sheetsides have entered the input queue of workstation j, then the output buffer of workstation j cannot be full, i.e., $\Delta_{out}\left[BQ_i^j\right] = 0$. In the second case, assume that more than N sheetsides have entered the input queue of workstation j, but at the time when sheetside BQ_i^j completes there will be at least one free slot in the output buffer of workstation j, i.e., at least sheetside BQ_{i-N}^j has left the output buffer, then $\Delta_{out}\left[BQ_i^j\right] = 0$. In the final case, if the output buffer of workstation j is full when sheetside BQ_{i-1}^j completes, then BQ_i^j must wait for an opening in the output buffer before its processing can begin. Therefore, sheetside BQ_i^j will be delayed until the sheetside at the head of the output buffer of the workstation completes transmission to the raster device. The three delay cases for BQ_i^j to begin processing can be described succinctly as follows:

```
if ( (size($S_k$) $\le AC^j_{in}$)   AND   ($|\mathcal{K}| < Q$ ))
  $t^j_{dept}[S_k] \leftarrow t^x_{dept}[S_{k-1}] + t^{sdf}_{tran}[S_{k-1}]$;
end if
else
  $BQ^j_i \leftarrow$ first element of $\mathcal{K}$;
  min_size $\leftarrow AC^j_{in}$;
  files $\leftarrow |\mathcal{K}|$;
  while ((size($S_k$) > min_size) OR (files $\ge Q$))
    min_size $\leftarrow$ min_size + size($BQ^j_i$);
    $BQ^j_i \leftarrow$ next element in $\mathcal{K}$;
    files $\leftarrow$ files – 1;
  end while
  $t^j_{dept}[S_k] \leftarrow t^j_{comp}[BQ^j_i]$
end else
```

Fig. 2. Pseudo-code for determining the earliest estimated departure time for sheetside S_k assigned to workstation j.

Case 1: if $i < N$,

$$\Delta_{out}\left[BQ^j_i\right] = 0.$$

Case 2: if $t_d[BQ_{i-N}] + t^{bitmap}_{tran} \leqslant t^j_{comp}\left[BQ^j_{i-1}\right]$,

$$\Delta_{out}\left[BQ^j_i\right] = 0.$$

Case 3: otherwise,

$$\Delta_{out}\left[BQ^j_i\right] = t_d\left[BQ^j_{i-N}\right] + t^{bitmap}_{tran} - t^j_{comp}\left[BQ^j_{i-1}\right]$$

Using $\Delta_{out}\left[BQ^j_i\right]$, we can define the estimated rasterization *start* time of sheetside BQ^j_i. That is, $t_{start}\left[BQ^j_i\right]$ occurs when two conditions are satisfied: BQ^j_i is present at the head of the input queue on workstation j, and the output buffer of workstation j has sufficient capacity to accommodate the rasterization result. If these conditions are not satisfied, then $t_{start}\left[BQ^j_i\right]$ is defined by one of two cases. First, if there is *no* opening in the output buffer of workstation j when BQ^j_{i-1} completes and BQ^j_i is available at the head of the input buffer of workstation j, then:

$$t_{start}\left[BQ^j_i\right] = t^j_{comp}\left[BQ^j_{i-1}\right] + \Delta_{out}\left[BQ^j_i\right]. \tag{5}$$

In the second case, if there is an opening in the output buffer of workstation j when BQ^j_{i-1} completes and BQ^j_i is *not* in the input buffer of workstation j, then the estimated start time of BQ^j_i is equal to the arrival time of BQ^j_i in the input buffer, i.e.:

$$t_{start}\left[BQ^j_i\right] = t_{dept}\left[BQ^j_i\right] + t^{sdf}_{tran}\left[BQ^j_i\right]. \tag{6}$$

That is, as soon as BQ^j_i arrives in the input buffer of workstation j, it will be rasterized without further delay. Note that if there is no opening in the output buffer on workstation j and BQ^j_i is not in the input buffer of workstation j, then one of the previous two cases will occur some time in the future. The two equations corresponding to the two cases for the earliest rasterization start time can be combined to calculate the estimated start time for BQ^j_i as follows:

$$t_{start}\left[BQ^j_i\right] = \max\left\{\left(t^j_{comp}\left[BQ^j_{i-1}\right] + \Delta_{out}\left[BQ^j_i\right]\right), \left(t_{dept}\left[BQ^j_i\right] + t^{sdf}_{tran}\left[BQ^j_i\right]\right)\right\}. \tag{7}$$

Note that calculation of $t^j_{comp}\left[BQ^j_i\right]$ is based on a recursion because it depends on $t_{start}\left[BQ^j_i\right]$ which in turn relies on $t^j_{comp}\left[BQ^j_{i-1}\right]$. The recursion basis is formed with BQ^j_1, whose $t^j_{comp}\left[BQ^j_1\right]$ is found as:

$$t^j_{comp}\left[BQ^j_1\right] = ERT\left[BQ^j_1\right] + t_{dept}\left[BQ^j_1\right] + t^{sdf}_{tran}\left[BQ^j_1\right]. \tag{8}$$

That is, because BQ^j_1 is the first sheetside to be rasterized on workstation j, there can be no delays incurred from processing earlier sheetsides on this workstation.

4. Minimum rasterization completion time heuristic (MRCT)

The resource allocation heuristic described in this section assumes that the system is in a steady state of operation, i.e., some sheetsides have already been rasterized and the start time t_0 for the display devices is known. In this situation, sheetsides are dispatched to the workstation that provides the minimum rasterization completion time as defined by our mathematical model of the system. That is, $t^j_{comp}[S_k]$ is calculated for every workstation j as if S_k were assigned to it, and the workstation that gives the minimum value is selected.

Because this is a dynamic environment, updates to the minimum rasterization completion time occur when rasterization completes for a given sheetside, i.e., the actual time required for rasterization becomes known. Immediately following a sheetside completion, a control message is sent to the head node informing it of the completion. The head node then updates the recurrence equation for calculating the completion time of any subsequent sheetsides with this new information, thus, the rasterization completion time estimates become more accurate. As a result of these updates, the minimum rasterization completion time workstation for a given sheetside may change over time, i.e., the heuristic choice is time dependent.

MRCT begins by calculating the $t^j_{comp}[S_k]$ values for the sheetside at the head of the head node input queue (S_k) for all of the workstations in the system. Workstations are then ranked in ascending order according to their $t^j_{comp}[S_k]$ values and placed together in a table in that order.

Afterwards, MRCT creates the table for ranking workstations for S_k. If there is room in the input buffer of the highest ranked workstation j, i.e., $AC^j_{in} \geqslant \text{size}(S_k)$ and $|\mathscr{K}| < Q$, and there is a free slot in the transmit queue, then S_k is assigned to the selected workstation and placed in the transmit queue. If any of the required conditions is not satisfied, then the conditions will be satisfied some time in the future. As each workstation completes a sheetside, the table for S_k is updated and reordered.

Because the execution time estimates for rasterization are known to be estimates of the actual execution times, the heuristic must account for cases where the estimated rasterization completion time for S_k is significantly under-estimated. For an under-estimated rasterization completion time to be significant, another workstation in the system must have a smaller or equivalent completion time for S_k compared to the actual rasterization completion time that has been under-estimated. The time at which the under-estimate for S_k becomes significant is referred to as the invalidation time for workstation j, denoted $INVT_j$.

To calculate $INVT_j$, the head node uses the rasterization completion notifications that it receives from each workstation. Because of this feedback, the head node can calculate the earliest expected feedback time ($EEFT_j$) for the completion of the sheetside currently being rasterized on workstation j, using the start time of the rasterization and the expected rasterization execution time. Using $EEFT_j$ and the estimated completion time for S_k on workstation (j) that is estimated to complete it soonest and the next best workstation (x), the invalidation time for a given workstation j ($INVT_j$) can be calculated as follows:

$$INVT_j = EEFT_j + \left(t^x_{comp}[S_k] - t^j_{comp}[S_k]\right). \tag{9}$$

That is, if at time $INVT_j$ the current sheetside being rasterized on workstation j has not completed, it has exceeded its estimated completion time ($EEFT_j$) by an amount ($t^x_{comp}[S_k] - t^j_{comp}[S_k]$) that now must make workstation x the best choice for S_k.

Once the ordering of workstations has been established, the $INVT_j$ values can be calculated for each of the workstations. If any of the $INVT_j$ values are in the past, then the corresponding workstations are "invalidated." The MRCT heuristic will not consider any workstations during allocation that are currently invalidated, i.e., while a workstation is marked as invalid no sheetsides will be assigned to it. When feedback regarding a sheetside completion on workstation j is received, the $EEFT_j$, $t^j_{comp}[S_i]$, and $INVT_j$ values for the associated workstation are recalculated and the invalidation status of the workstation is reset to valid. The MRCT heuristic is summarized in Fig. 3.

```
for all $j$
    calculate $t^j_{comp}[S_k]$;
    calculate $EEFT_j$;
end for
sort workstations in ascending order of $t^j_{comp}[S_k]$;
calculate $INVT_j$ for all $j$;
min ← $argmax_{\forall j} \{t^j_{comp}[S_k]\}$;
for all $j$
    if $((t^j_{comp}[S_k] < t^{\min}_{comp}[S_k])$ AND $(INVT_j > t))$
      min ← $j$;
    end if
if $\left(AC^{IN}_{\min} < \text{size}(S_k))\right.$ AND $\left.|TQ| < 2\right)$
   assign $S_k$ to min;
end if
else
    wait;
end else
```

Fig. 3. Pseudo-code for assigning sheetside S_k at the head of HNIQ at time t in the MRCT heuristic.

5. Bitmap lifetime

In general, the primary goal of the system is to ensure that all incoming sheetsides are rasterized and available by their deadline for display. To assess whether a sheetside is available by its deadline, we defined a new measure known as “bitmap lifetime.” *Bitmap lifetime* is measured as the time difference between when the rasterized image is made available in some workstation’s output buffer and when the raster display consumes the image from the system, i.e., the amount of time that a bitmap *lives* in an output buffer of the system before it is displayed.

When the bitmap lifetime is greater than t_{tran}^{bitmap}, the bitmap will arrive in time to be displayed without disrupting the system. If lifetime of a bitmap is not greater than t_{tran}^{bitmap}, then the bitmap will not arrive in time to be displayed by its deadline, and the system is disrupted. To represent this in our simulations, we say that whenever a bitmap lifetime is not greater than t_{tran}^{bitmap}, it is given the value t_{tran}^{bitmap} and will cause a system disruption.

The MRCT heuristic attempts to minimize the rasterization completion time of a sheetside S_k based on its most current estimates of the system state. Alternatively, we can view this minimization as an attempt to maximize the bitmap lifetime of S_k. Because the deadline for the completion of each sheetside is fixed relative to t_0, by minimizing the estimated completion time for each sheetside we are maximizing the difference between the deadline for a sheetside and its completion time.

6. Quantifying robustness

6.1. Overall robustness metric

To derive a robustness metric suitable for this environment, we follow the FePIA procedure presented in [1]. In step 1 of the FePIA procedure, we describe quantitatively the requirement that makes the system robust. Intuitively, the system is robust if no service interruptions occur. That is, the performance measure of interest in this system is the completion time of each sheetside. If the completion time for each sheetside is less than its deadline, then the system can be considered to be robust.

In step 2, we identify the uncertainty in system parameters that may impact our performance feature of interest. In this system, there are two sources of uncertainty in system parameters that may cause our rasterization completion times to increase (possibly violating our robustness requirement). First, we have assumed throughout this work that our rasterization execution time estimates may differ substantially from actual rasterization execution times. Second, the arrival order of sheetsides to the system is unknown. That is, we do not know in advance when complex sheetsides will arrive for rasterization.

Step 3 of the FePIA procedure requires that we identify the impact of uncertainty in system parameters on our performance feature of interest. In this case, rasterization completion time estimates are created based on a sum of rasterization execution estimates that each may contain errors. Consequently, the uncertainty in rasterization execution times will directly impact rasterization completion time estimates.

The last step is to conduct an analysis to determine the smallest collective change in assumed values for system uncertainty parameters (from step 2) that would cause the performance feature of step 1 to violate the robustness requirement. To determine the robustness of an overall resource allocation, we will first quantify the robustness of the completion time estimate for a single sheetside BQ_i^j. Let $\mathscr{B}_i^j$ be the set of sheetsides pending on workstation j before and including BQ_i^j. The rasterization execution time estimates for any of the sheetsides in $\mathscr{B}_i^j$ may be a source of uncertainty in calculating the rasterization completion time estimate for BQ_i^j. The completion time estimates for these sheetsides are coupled because they are executed sequentially on the same workstation. That is, if the rasterization time of the first sheetside is longer than expected, then this will impact the completion time calculations for each of the subsequent sheetsides on that workstation.

From [1], we can use a geometric interpretation of our robustness requirement where the rasterization completion time estimate is a single point in an $\mathscr{N}$-dimensional space. Each of the $\mathscr{N}$ dimensions in this space corresponds to a member of $\mathscr{B}_i^j$ and we wish to find the smallest distance from our rasterization completion time estimate to the surface defined by our robustness requirement. This distance defines the smallest collective increase in assumed system parameters that would cause our robustness requirement to be violated (based on a Euclidean distance). Because the completion time estimates for each sheetside are coupled, the true geometric interpretation of the robustness requirement must be expressed as an $\mathscr{N}$ dimensional surface. Without knowing the exact shape of this surface, we cannot calculate the shortest distance from our known point to the surface. Thus, to calculate this distance in our Robust MRCT heuristic (presented in Section 6.2, we have chosen to approximate this surface by a hyperplane. Using the equation for the distance from a point to a hyperplane [1] we can find the robustness of the completion time for sheetside BQ_i^j given a current assignment of sheetsides μ at time t, denoted $r_\mu(BQ_i^j, t)$ as:

$$r_\mu\left(BQ_i^j, t\right) = \frac{t_d\left[BQ_i^j\right] - t_{comp}^j\left[BQ_i^j\right]}{\sqrt{\mathscr{B}_i^j}}. \tag{10}$$

To find the overall robustness of a resource allocation at time t, we identify the smallest robustness value for each workstation and then compare this smallest value across all workstations. We combine the $r_\mu(BQ_i^j, t)$ values for all $\mathscr{B}_i^j$ at time t to form the robustness metric for workstation j at time t, denoted $\rho_\mu^j(t)$, as follows:

$$\rho_{\mu}^{j}(t) = \min_{\forall i \in \mathscr{B}_i^j} \left\{ r_{\mu}\left(BQ_i^j, t\right) \right\}. \tag{11}$$

The smallest of the $\rho_{\mu}^{j}(t)$ values over all workstations defines the *local* robustness value for the system at time t. That is,

$$\rho_{\mu}(t) = \min_{\forall j} \left\{ \rho_{\mu}^{j}(t) \right\}. \tag{12}$$

Because any service interruption is unacceptable in this environment, we have chosen to use the smallest local robustness value encountered throughout a simulation run as the *overall* robustness metric. Formally, we can express the *overall robustness metric* value for a particular resource allocation in this system as:

$$\rho_{\mu} = \min_{\forall t} \left\{ \rho_{\mu}(t) \right\}. \tag{13}$$

6.2. Robust MRCT

The robustness of each sheetside completion time estimate can be used during resource allocation to aid in selecting the best workstation to process a given sheetside. In the MRCT heuristic presented earlier, we can replace the rasterization completion estimate determined for each workstation with the robustness metric.

If we compare the bitmap lifetime metric with robustness, then we can see that the numerator in the robustness Eq. (10) is actually an estimate of bitmap lifetime at time t. The fundamental difference between the two calculations is the denominator of the robustness equation that attempts to account for the multiple uncertainty parameters in the bitmap lifetime estimate. However, in this environment, because the number of sheetsides that can be buffered in the input queue of each workstation is limited, the magnitude of the denominator in the robustness equation is also limited.

7. Simulation setup

To evaluate our heuristics, we created a simulation model of a real printing system using the OpNet simulation environment [16]. Each simulation run consisted of a rasterization job that included on the order of 100,000 sheetsides. The OpNet simulation was executed with a head node connected by a gigabit ethernet network to six heterogeneous workstations. The workstations are connected to the two raster display devices by a four gigabit fiber channel. It is assumed that the raster display device requires 0.11 s to display each output bitmap.

Each sheetside rasterization time estimate is modeled by sampling one of two normal distributions. The first distribution was chosen to have a mean of 0.01 s and a standard deviation of 20% of the mean. The second distribution was chosen to have a mean of 0.85 s and a standard deviation of 20% of the mean. In our simulations, the ratio of 0.85 s sheetsides to 0.01 s sheetsides was chosen such that the average rasterization time was 0.22 s. This average rasterization time was chosen to match the processing time required to output a single bitmap from each display device. Using the chosen ratio of sheetsides, for every rasterization time sampled from the 0.85 s mean distribution there are three sheetsides selected from the 0.01 s mean distribution.

For comparison, we implemented a round-robin heuristic that was run on identical simulations to that of our MRCT heuristic and Robust MRCT heuristic. *Round-robin* tries to assign the same number of sheetsides to each workstation in the cluster by defining an arbitrary fixed ordering of the workstations and repeatedly assigning one sheetside to each workstation in the ordering as buffer sizes permit [22]. If there is insufficient capacity in the input buffer of any workstation j or there are greater than Q sheetsides in the input buffer already, then round-robin waits until both of these conditions are satisfied on workstation j so that the machine ordering is obeyed. Consequently, round-robin ignores the current workload on each of the workstations, instead relying on a strict “balanced” ordering of sheetside assignments to fairly assign the workload among machines.

Although the simulation study did not attempt to directly evaluate a startup strategy for starting the displays, the simulation required some startup to begin execution. For this simulation study, we chose a simplistic strategy where the sheetsides are allocated to workstations in a round-robin fashion, prior to starting the displays, until all of the workstation output buffers are full. At this point, the displays are turned on and the centralized image dispatcher begins to use one of the three studied heuristics to allocate the remaining sheetsides for the remainder of the simulation: Round-Robin, MRCT, or Robust MRCT.

8. Simulation results

Fig. 4 presents the results of the simulation study in terms of bitmap lifetime. The x axis of each plot represents the simulation time in seconds and the y axis represents the bitmap lifetime in seconds. The plots show the bitmap lifetime values for each bitmap consumed by the system. As described in Section 5, if the lifetime of a bitmap is not greater than t_{tran}^{bitmap}, then the display device will have to stop to wait for the bitmap to become available—which is unacceptable in practice. In a large scale production printing environment, the paper where the raster device is displaying the images cannot be immediately

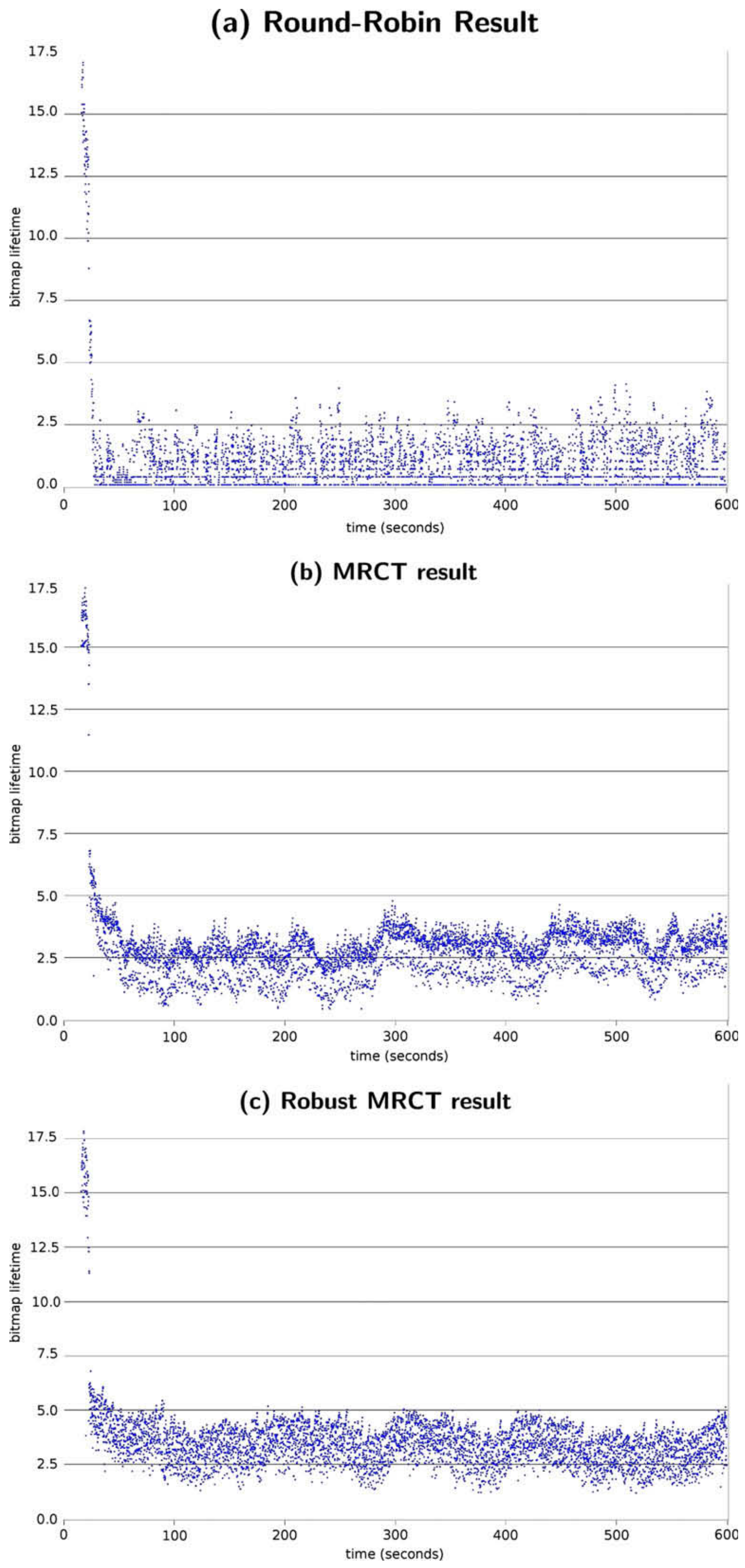

Fig. 4. Sample plots of the results for the three heuristics (a) round-robin, (b) MRCT, and (c) Robust MRCT.

stopped to wait for bitmaps to become available. Attempting to abruptly stop the paper may ruin the result, e.g., by tearing the paper.

In each of the plots of Fig. 4, at the beginning of each simulation the bitmap lifetimes are high relative to the mean bitmap lifetime. These artificially high values occur before t_0, i.e., during this time the displays have not started to consume bitmaps. Thus, the initial bitmap lifetimes are equal to the time required to fill up the output buffers on all of the workstations prior to starting the display device.

The simulation required that each heuristic rasterize the same number of sheetsides on the same set of workstations (four in this study), where the output was consumed by two displays. The round-robin heuristic is able to complete the entire run in only slightly more time than both of the MRCT heuristics, however, it did experience a significant number of service interruptions as a result of its allocation decisions. Recall that in a high-speed printing environment any interruption is considered catastrophic. In contrast, the MRCT heuristic based on bitmap lifetime is able to complete the entire run with no interruptions. For the MRCT heuristic, bitmap lifetimes were in the range of $[1.6\text{ s}, 16\text{ s}]$ throughout the simulation. These results demonstrate the utility of the mathematical model to estimate rasterization completion times within the context of a resource allocation heuristic.

Finally, the results of our Robust MRCT heuristic surpass that of the bitmap lifetime based MRCT heuristic, as can be seen by comparing the plots of the two heuristic results. That is, the mean bitmap lifetime for sheetsides in the Robust MRCT heuristic is higher than that of the MRCT heuristic. This implies, that the Robust MRCT heuristic is capable of tolerating more additional complex sheetsides without interrupting service than the bitmap lifetime based MRCT heuristic. In our simulations, the time required to calculate the robustness value during execution of the Robust MRCT heuristic was negligible. Because the Robust MRCT heuristic uses the same rasterization completion time model as MRCT, the improvements of Robust MRCT can be solely attributed to the use of robustness in the heuristic in place of bitmap lifetime.

9. Related work

According to the literature, the problem of workload distribution considered in our research falls into the category of dynamic resource allocation, assuming that multiple invocations of a resource allocation heuristic are overlapped in time with task arrivals. The general problem of dynamically allocating a class of independent tasks onto heterogeneous computing systems was studied in [13]. The primary objective in [13] was to minimize system makespan, i.e., the total time required to complete all tasks sent for mapping. This objective is very different from the primary objective in our work: complete rasterization of each sheetside in a given job before its assigned deadline. Our MRCT heuristic attempts to map each sheetside to its estimated minimum rasterization completion time workstation, which is analogous to the MCT heuristic of [13] attempting to map each task to its minimum completion time machine. However, the method of computing a completion time in [13] does not take into account the impacts of buffering tasks, communication links, etc. Furthermore, that study assumes no deviation of the actual time to compute a task from its estimated time to compute (ETC) value, i.e., the performance predicted by a resource allocation heuristic is assumed to match the actual performance. In our simulations, the heuristics are provided with estimated execution times that can differ from the ‘‘simulated actual” execution times. In our MRCT approach, rasterization completion time estimates for a sheetside are continuously updated with the most current information regarding the ‘‘simulated actual” sheetside completion times.

In [14], an end-to-end quality of service system is described for a distributed real-time embedded system. The authors define quality of service within an embedded system loosely as, ‘‘how well an application performs its function.” The authors advocate an approach where resource allocation decisions are dynamically adapted to changes in the environment based on the coordinated monitoring and control of constrained system resources. Our work is an application of this methodology within a specific real-time imaging system. In the terminology of [14], the resource management techniques of our research are appropriate for use in the System Resource Manager role of the multi-layer resource management architecture.

In [10], a number of resource allocation heuristics for a class of independent tasks were tested on a homogeneous cluster of eight DEC Alpha workstations running Digital Unix. The set of presented heuristics includes the following five: round-robin; round-robin with clustering; minimal adaptive; continual adaptive; and first-come first-served. None of these heuristics built a prediction model.

The robustness requirement in this work differs substantially from our earlier work on robustness in a dynamic environment [15]. In [15], the robustness requirement was expressed in terms of the overall resource allocation, i.e., expressed in terms of the entire allocation. In this work, each sheetside has an individual deadline, thus, the robustness metric must be expressed in terms of individual sheetsides. In [20], each dynamically arriving task is assigned its own deadline relative to its arrival time. However, that work assumes that stochastic information is available regarding the possible execution times of tasks. In this environment, we are only provided with a deterministic estimate of task execution times and this stochastic information is unavailable.

10. Conclusion

The goal of this research was to rasterize dynamically arriving sheetsides (i.e., execute tasks) before an assigned deadline for each sheetside so that service interruptions can be avoided during execution. We presented a mathematical model suitable for determining an estimate of rasterization completion times in a dynamic environment where task execution times are uncertain. We used the mathematical model to design the MRCT resource management heuristics that clearly outperformed a commonly used approach, i.e., round-robin. Further, this mathematical model was used as the basis for deriving a robustness metric suitable for this environment. We presented an extension of our MRCT heuristic, Robust MRCT, that successfully utilized this robustness metric during resource allocation to surpass the results of our bitmap lifetime based approach.

Acknowledgement

A preliminary version of a portion of this work was presented at the Parallel and Distributed Processing Techniques and Applications conference in March 2007. There is a patent pending for the application of the MRCT technique within the context of a printing environment.

Appendix.

Glossary of notation.

AC^j_{in}	Available capacity of the input buffer of workstation j
$\mathscr{B}^j_i$	Set of sheetsides pending on workstation j
BQ^j_i	ith sheetside to have entered the input queue of workstation j
CAP^j_{in}	Input buffer capacity of workstation j
CAP^j_{out}	Output buffer capacity of workstation j
$\Delta_{out}\left[BQ^j_i\right]$	Delay to begin processing sheetside BQ^j_i
$EEFT_j$	Earliest expected feedback time for workstation j
$ERT\left[BQ^j_i\right]$	Estimated rasterization time for sheetside BQ^j_i
HNIQ	Head node input queue
$INVT_j$	Invalidation time for workstation j
$\mathscr{K}$	Sequence of sheetsides that are in the input buffer of workstation j
M	Number of heterogeneous dedicated workstations
μ	Current allocation of sheetsides to workstations
N	Number of bitmaps that can be placed in the output buffer of a given workstation
$\mathrm{num}(BQ^j_i)$	Absolute sheetside number of sheetside BQ^j_i, i.e., S_k
Q	Maximum number of sheetsides allowed in an input buffer
$r_\mu(BQ^j_i, t)$	Robustness of the completion time for sheetside BQ^j_i given assignment μ at time t
ρ_μ	Overall robustness metric value given assignment μ
$\rho^j_\mu(t)$	Robustness metric for workstation j given assignment μ at time t
S_k	kth sheetside of the job
$\mathrm{size}(S_i)$	Size in bytes of the pdl for sheetside i
t_0	Absolute wall-clock start time for both display devices
t^{bitmap}_{tran}	Transfer time from any workstation to either display device
$t^j_{comp}\left[BQ^j_i\right]$	Estimated rasterization completion time for sheetside BQ^j_i on workstation j
$t_d[S_k]$	Wall-clock deadline for completing sheetside S_k
$t^x_{dept}[S_{k-1}]$	Departure time of S_{k-1} from HNIQ to workstation x
$t_{display}$	Time required to display a bitmap on the device
$t_{start}\left[BQ^j_i\right]$	Earliest possible rasterization start time for sheetside BQ^j_i
$t^{sdf}_{tran}[S_k]$	Time required to transfer the sheetside description file for sheetside S_k
$\|TQ\|$	Number of sheetsides in head node transmitter queue

References

[1] S. Ali, A.A. Maciejewski, H.J. Siegel, J.-K. Kim, Measuring the robustness of a resource allocation, IEEE Transactions on Parallel and Distributed Systems 15 (7) (2004) 630–641.

[2] S. Ali, A.A. Maciejewski, H.J. Siegel, Perspectives on Robust Resource Allocation for Heterogeneous Parallel Systems, Chapman & Hall/CRC Press, Boca Raton, FL, 2008. pp. 41-1–41-30.

[3] I. Banicescu, V. Velusamy, Performance of scheduling scientific applications with adaptive weighted factoring, in: 10th IEEE Heterogeneous Computing Workshop (HCW 2001), in the Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001), April 2001.
[4] E.G. Coffman (Ed.), Computer and Job-Shop Scheduling Theory, John Wiley & Sons, New York, NY, 1976.
[5] D. Fernandez-Baca, Allocating modules to processors in a distributed system, IEEE Transactions on Software Engineering 15 (11) (1989) 1427–1436.
[6] I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufman, San Francisco, CA, 1999.
[7] A. Ghafoor, J. Yang, A distributed heterogeneous supercomputing management system, IEEE Computer 26 (6) (1993) 78–86.
[8] N. Gharachorloo, S. Gupta, R.F. Sproull, I.E. Sutherland, A characterization of ten rasterization techniques, in: SIGGRAPH'89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press, New York, NY, USA, 1989, pp. 355–368.
[9] O.H. Ibarra, C.E. Kim, Heuristic algorithms for scheduling independent tasks on non-identical processors, Journal of the ACM 24 (2) (1977) 280–289.
[10] H.A. James, K.A. Hawik, P.D. Coddington, Scheduling independent tasks on metacomputing systems, in: Proceedings of the 12th International Conference on Parallel and Distributed Computing Systems (PDCS 99), August 1999, pp. 47–52.
[11] M. Kafil, I. Ahmad, Optimal task assignment in heterogeneous distributed computing systems, IEEE Concurrency 6 (3) (1998) 42–51.
[12] C. Leangsuksun, J. Potter, S. Scott, Dynamic task mapping algorithms for a distributed heterogeneous computing environment, in: Proceedings of the Fourth IEEE Heterogeneous Computing Workshop (HCW'95), April 1995, pp. 30–34.
[13] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, Journal of Parallel and Distributed Computing 59 (2) (1999) 107–121.
[14] P. Manghwani, J. Loyall, P. Sharma, M. Gillen, J. Ye, End-to-end quality of service management for distributed real-time embedded applications, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), April 2005.
[15] A.M. Mehta, J. Smith, H.J. Siegel, A.A. Maciejewski, A. Jayaseelan, B. Ye, Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness, Journal of Supercomputing 42 (1) (2007) 33–58. special issue on Grid Technology.
[16] Opnet Technologies Inc., 2007. Available [online] <http://www.opnet.com/> (accessed 20.2.07).
[17] V. Shestak, E.K.P. Chong, H.J. Siegel, A.A. Maciejewski, L. Benmohamed, I. Wang, R. Daley, A hybrid branch-and-bound and evolutionary approach for allocating strings of applications to heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing 68 (4) (2008) 410–426.
[18] V. Shestak, J. Smith, A.A. Maciejewski, H.J. Siegel, Stochastic robustness metric and its use for static resource allocations, Journal of Parallel and Distributed Computing 68 (8) (2008) 1157–1173.
[19] H. Singh, A. Youssef, Mapping and scheduling heterogeneous task graphs using genetic algorithms, in: Proceedings of the Fifth IEEE Heterogeneous Computing Workshop (HCW '96), 1996, pp. 86–97.
[20] J. Smith, L.D. Briceño, A.A. Maciejewski, H.J. Siegel, Measuring the robustness of resource allocations in a stochastic dynamic environment, in: Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007), March 2007.
[21] P. Sugavanam, H.J. Siegel, A.A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, A. Pippin, Robust static allocation of resources for independent tasks under makespan and dollar cost constraints, Journal of Parallel and Distributed Computing 67 (4) (2007) 400–416.
[22] X. Tang, S.T. Chanson, Optimizing static job scheduling in a network of heterogeneous computers, in: Proceedings of the International Conference on Parallel Processing 2000 (ICPP'00), August 2000, p. 373.
[23] M. Wu, W. Shu, H. Zhang, Segmented min–min: a static mapping algorithm for meta-tasks on heterogeneous computing systems, in: Proceedings of the Ninth IEEE Heterogeneous Computing Workshop, March 2000, pp. 375–385.
[24] D. Xu, K. Nahrstedt, D. Wichadakul, Qos and contention-aware multi-resource reservation, Cluster Computing 4 (2) (2001) 95–107.