Resource Allocation in a Cluster Based Imaging System

Jay Smith[†*], Vladimir Shestak[†*], Suzy Price[†], and Larry Teklits[†]

[†]IBM
Boulder, CO 80301
Email: {bigfun, vshestak, suzyp, larryt}@us.ibm.com

Howard Jay Siegel[*‡] and Prasanna Sugavanum[*]

[*]Electrical and Computer Engineering Department
[‡]Computer Science Department
Colorado State University
Fort Collins, CO 80523–1373
Email: {hj, psugavanum}@engr.colostate.edu

***Abstract*—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. We present a mathematical model of such a cluster based raster imaging system. 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 output 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 design a resource allocation heuristic capable of completing each rasterization task before its deadline, thus, preventing any service interruptions. This new heuristic is evaluated within a simulation of the studied raster imaging system. We clearly demonstrate the effectiveness of the heuristic by comparing its results with the results of two resource allocation heuristics commonly used in this type of system.**

***Index Terms*—heterogeneous computing, resource management, dynamic resource allocation, distributed computing**

I. Introduction

Recently there has been an increased demand for imaging systems in support of high-speed digital printing. The required increase in performance can be accomplished 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 regular 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 a given distributed computing environment. The new heuristic is then evaluated within the context of a simulation of the raster imaging system, where the results of the heuristic are compared with the results of several heuristics commonly used in systems of this type.

The primary contributions of this work are a mathematical model of a dynamic distributed computing system with hard deadlines on task execution times and an application of this model to the design of a resource allocation heuristic suitable for this type of system. In addition, we clearly demonstrate the heuristic's superiority over two common resource management techniques.

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, arrives at an imaging system for rasterization [8]. 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 cluster by a centralized image dispatcher (CID). 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 today's fastest 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 rendered images, referred to as bitmaps. The bitmaps to be displayed are retrieved at a regular interval directly from the system output buffer by the display device. By retrieving the first bitmap from an output buffer, a hard deadline for each subsequent bitmap is established. Missing a deadline for a required bitmap results in an interruption of the display device that is unacceptable.

The studied raster image processing (RIP) system has some additional special requirements that complicate the task of assigning the stream of incoming pdl images to available workstations. First, the system only has an estimate of the time required to rasterize each incoming pdl image and this estimate may differ subtantially 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 that constrains the number of pdl images that can be rasterized in advance of the image stream being consumed by the display device. Hence, 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, the pdl images to be rasterized are continually arriving in parallel with their consumption, i.e., the resource allocation must be produced dynamically [14]. 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, dynamic resource allocation is an active area of research [2], [6], [14], [15], [21], [23].

The concept of robustness of a resource allocation was introduced in [1], [18], and later efforts applied the concept to resource management [11], [15], [17], [19], [21]. This research did not require an explicit measure of robustness, however, the developed mathematical model does incorporate many of the same basic principals addressed in our earlier work on robustness. In analyzing the image processing system, we quickly identified the source of uncertainty in the system, i.e., that rasterization times are uncertain and the arrival ordering of sheetsides is unknown. 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. Therefore, a simplistic quantification robustness in this system can be defined as the difference in time between when a bitmap is made available in an output buffer and when that bitmap is retrieved for display. In reviewing our results, we compare the three studied heuristics using a measure of this difference referred to as bitmap lifetime, where a robust heuristic always maintains a bitmap lifetime that is greater than 0.

In the next section, we present the details of the system model used to design the mathematical model of RIP completion times presented in Section III. The new resource allocation heuristic that incorporates this mathematical model for RIP completion times is presented in Section IV. Section V presents the details of the simulation setup used to evaluate the heuristic and the results are presented in Section VI. A sampling of related work is in Section VII.

II. System Model

In this environment, a collection of workstations dedicated to image rasterization are controlled by a single computer, known as the $\underline{\text{head}}$ $\underline{\text{node}}$. Individual pdl images, referred to as sheetsides, are transferred to the head node. Each sheetside completely describes an entire display image suitable for the output device.

Input sheetsides are queued for rasterization in the $\underline{\text{H}}$ead $\underline{\text{N}}$ode $\underline{\text{I}}$nput $\underline{\text{Q}}$ueue ($\underline{\text{HNIQ}}$). The CID, in the head node, assigns sheetsides to workstations for rasterization. After assignment, the head node places the sheetside in a queue for a transmitter that will then transmit the data to its destination. The size of the transmitter queue is limited to only two sheetsides and it may only transfer one sheetside at a time. Also, once a sheetside has been placed in the transfer queue, the sheetside's destination workstation can no longer be modified. Each workstation has a finite input buffer capacity for storing sheetsides prior to rasterization. Therefore, before the incoming sheetside can be transfered, the head node must ensure that sufficient capacity exists in the input buffer of the receiving workstation to receive the file. If there is sufficient input buffer capacity, then the sheetside may be queued up for transfer.

It is assumed that $\underline{M}$ heterogeneous dedicated workstations are available to convert pdl sheetsides to bitmaps. Each workstation is interconnected to the head node via a 1 Gbit/second ethernet network and interconnected to the *two* output display devices using a 4Gbit/second 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 rasterization of a sheetside, notification is sent to the head node, and the appropriate display device, indicating the sheetside completion. The display device retrieves completed bitmaps for display from the output buffer of the workstation where the rasterization was performed. The input buffer of the display device has sufficient capacity to store two bitmaps, i.e., the bitmap currently being displayed and the next bitmap to be displayed. In this system, the size of the bitmaps is constant and the time required to display the bitmap is also constant.

III. Model of RIP Completion Time

This mathematical model defines a means for calculating the deadline for a given sheetside, as well as determining the estimated RIP completion time for a given sheetside. Recall that every sheetside is subject to a hard deadline. To prevent a service interruption, the sheetside must have completed rasterization, and must be available for consumption in the input buffer of the display device by that deadline. To calculate the hard deadline for a sheetside, let $\underline{t_0}$ be the absolute wall-clock start time for both display devices. Starting at t_0, each display will require a new bitmap every $\underline{t_{display}}$ seconds, where $t_{display}$ is the time required to display a bitmap. Sheetsides are numbered starting with 1. For the k^{th} actual sheetside of the job, denoted $\underline{S_k}$, the bitmap has to 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. Let $\underline{t_{tran}^{bitmap}}$ be the bitmap transfer time from any workstation to either display device. Then, S_k's $\underline{\text{deadline}}$ $\underline{t_d[S_k]}$ is the latest wall-clock time for a workstation to produce S_k's bitmap. 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.

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

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 $\underline{BQ_i^j}$ be the i^{th} sheetside to have entered workstation j's input queue for a given job. Define the operator $\underline{\text{num}(BQ_i^j)}$ that evaluates to the actual sheetside number, i.e., $S_k = \text{num}(BQ_i^j)$. Then S_k in Equation 1 can be replaced by $\text{num}(BQ_i^j)$. Note that S_k and BQ_i^j represent the same physical sheetdside and their notations will be used interchangeably.

The estimated RIP *completion* time for a sheetside is composed of the earliest possible time that rasterization can begin and the estimated rasterization time. Let $\underline{t_{comp}[BQ_i^j]}$ be the estimated completion time for a given sheetside BQ_i^j on workstation j, let $\underline{t_{start}[BQ_i^j]}$ be the earliest possible time that rasterization can begin, and let $\underline{ERT[BQ_i^j]}$ be the estimated rasterization time for sheetside BQ_i^j. Then $t_{comp}[BQ_i^j]$ can be calculated as,

$$t_{comp}[BQ_i^j] = t_{start}[BQ_i^j] + ERT[BQ_i^j]. \quad (2)$$

The estimated rasterization time, $ERT[BQ_i^j]$, for each sheetside is assumed known based on empirical data. There are many well-known techniques for gathering execution time estimates [7], [12], [13], [20], [24]. Therefore, calculating the estimated completion time for a sheetside amounts to determining the earliest possible time that the sheetside can begin rasterization. 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 workstation's output buffer.

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 k^{th} sheetside to enter the HNIQ for a given job, where S_{k-1} is the sheetside ahead of S_k in the HNIQ. Then 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 ($\underline{Q}$) allowed in the input buffer, and the size in bytes of the buffer.

Recall that the estimated rasterization times are known to be inaccurate relative to actual rasterization times; therefore, 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. This limitation attempts 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 workstation j's input buffer, S_k can be immediately sent to j following the transfer of S_{k-1} out of the head node. Otherwise, S_k will be delayed at the head node 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 accomodate 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 $\mathcal{K}$ be the sequence of sheetsides that are in workstation j's input buffer 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. Given the operator $\underline{\text{size}(S_k)}$ that gives the size of sheetside S_k's pdl file in bytes and let $\underline{CAP_{in}^j}$ describe the total input buffer capacity of workstation j. Then, the available capacity of workstation j's input buffer, denoted $\underline{AC_{in}^j}$, is

$$AC_{in}^j = CAP_{in}^j - \sum_{\forall S_k \in \mathcal{K}} \text{size}(S_k). \quad (3)$$

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

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

Otherwise, S_k must wait for a sufficient number of sheetsides to be processed from workstation j's input queue, to ensure that these two conditions hold. If after processing some sheetside $S_m \in \mathcal{K}$ 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 CID, then S_k can depart immediately.

Prior to rasterization, accurately determining when rasterization can begin for some sheetside BQ_i^j, where $S_k = num(BQ_i^j)$, 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 $\underline{CAP^j_{out}}$. Because bitmaps are all assumed to be the same size, i.e., require the same number of bytes, 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^j_i, denoted $\underline{\Delta_{out}[BQ^j_i]}$, as the time that BQ^j_i must wait after arriving at the head of the workstation's input queue until there is sufficient capacity in the output buffer of the workstation to store the output bitmap. To quantitatively determine $\Delta_{out}[BQ^j_i]$, there are three cases to consider. First, if fewer than N sheetsides have entered workstation j's input queue, then the output buffer of workstation j cannot be full, i.e., $\Delta_{out}[BQ^j_i] = 0$. In the second case, assume that more than N sheetsides have entered workstation j's input queue, but at the time when sheetside BQ^j_i completes there will be at least one free slot in workstation j's output buffer, i.e., at least sheetside BQ^j_{i-N} has left the output buffer, then $\Delta_{out}[BQ^j_i] = 0$. In the final case, if the output buffer of workstation j is full when sheetside BQ^j_{i-1} completes, then BQ^j_i must wait for an opening in the output buffer before its processing can begin. Therefore, sheetside BQ^j_i will be delayed until the sheetside at the head of the workstation's output buffer completes transmission to the raster device. The three delay cases for BQ^j_i to begin processing can be described succinctly as follows.

(case 1) if $i < N$,

$$\Delta_{out}[BQ^j_i] = 0 \quad (5)$$

(case 2) if $t_d[BQ_{i-N}] + t^{bitmap}_{tran} \le t_{comp}[BQ^j_{i-1}]$,

$$\Delta_{out}[BQ^j_i] = 0 \quad (6)$$

(case 3) otherwise,

$$\Delta_{out}[BQ^j_i] = t_d[BQ^j_{i-N}] + t^{bitmap}_{tran} - t_{comp}[BQ^j_{i-1}] \quad (7)$$

Using $\Delta_{out}[BQ^j_i]$, we can define the estimated rasterization *start* time of sheetside BQ^j_i. That is, $t_{start}[BQ^j_i]$ occurs when two conditions are satisfied: BQ^j_i is present at the head of the input queue on workstation j, and workstation j's output buffer has sufficient capacity to accomodate the rasterization result. If these conditions are not satisfied, then $t_{start}[BQ^j_i]$ 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

$$t_{start}[BQ^j_i] = t_{comp}[BQ^j_{i-1}] + \Delta_{out}[BQ^j_i]. \quad (8)$$

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.,

$$t_{start}[BQ^j_i] = t_{dept}[BQ^j_i] + t^{sdf}_{tran}[BQ^j_i]. \quad (9)$$

That is, as soon as BQ^j_i arrives in the input buffer, 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,

$$t_{start}[BQ^j_i] = \max\{\Big(t_{comp}[BQ^j_{i-1}] + \Delta_{out}[BQ^j_i]\Big),\ \Big(t_{dept}[BQ^j_i] + t^{sdf}_{tran}[BQ^j_i]\Big)\}. \quad (10)$$

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

$$t_{comp}[BQ^j_1] = ERT[BQ^j_1] + t_{dept}[BQ^j_1] + t^{sdf}_{tran}[BQ^j_1]. \quad (11)$$

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.

IV. Minimum RIP 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 RIP completion time as defined in the previous section. That is, $t_{comp}[S_k]$ is calculated for every workstation j, and the workstation that gives the minimum value is selected. Because this is a dynamic environment where sheetsides are arriving for rasterization while rasterization of other sheetsides is completing, the minimum RIP completion time workstation is time dependent. When rasterization is complete for a given sheetside, the actual time required for rasterization becomes known, and 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 RIP completion time estimates become more accurate. Consequently, the minimum RIP completion time workstation for a given sheetside may change during execution.

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

Recall, when a workstation completes rasterizing a sheetside a notification is sent to the head node to inform it of the completion. Because of this feedback, the head node can calculate the earliest expected feedback time ($\underline{EEFT_j}$) 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 of the best workstation (j) and the next best workstation (x), an invalidation time for a given workstation j can be calculated as follows,

$$INVT_j = EEFT_j + \left(t^x_{comp}[S_k] - t^j_{comp}[S_k]\right). \quad (12)$$

MRCT begins by calculating the $t^j_{comp}[S_k]$ values for sheetside S_k for all of the workstations in the system, where S_k is assumed to be at the head of the head node input queue. The workstations are then ranked in ascending order according to their $t^j_{comp}[S_k]$ value in a table. Once the ordering 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 invalidated workstations during allocation, i.e., while a workstation is marked as invalid no sheetsides will be assigned to it. When feedback regarding a sheetside completion on a workstation is received, the $EEFT_j$, $t^j_{comp}[S_i]$, and $INVT_j$ values are recalculated and the invalidation status of the workstation is reset to valid.

After MRCT creates the table, if there is room in the input buffer of the highest ranked workstation, i.e., $AC^j_{in} >$ size(S_i) and $|\mathcal{K}| < Q$, and there is a free slot in the transfer queue, then S_i is assigned to the selected workstation and placed in the transfer 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.

V. Simulation Setup

To evaluate the heuristic, we created a simulation model of the real system and executed the MRCT heuristic using jobs that included on the order of 100,000 sheetsides. The simulation consisted of a head node connected by a gigabit ethernet network to four workstations used to process the incoming jobs. 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 seconds to display each output bitmap. The mean sheetside rasterization time is assumed to be 0.22 seconds. The simulation was developed using the opNet simulation environment [16].

For comparison, we also implemented a round-robin heuristic and a random assignment 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 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 workstations, instead relying on a strict ordering of sheetside assignments to "balance" the workload among machines. The random assignment heuristic instead randomly assigns sheetsides to workstations with no regard to workstation workloads or sheetside assignment order [22].

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 CID begins to use one of the three studied heuristics to allocate the remaining sheetsides for the remainder of the simulation.

VI. Simulation Results

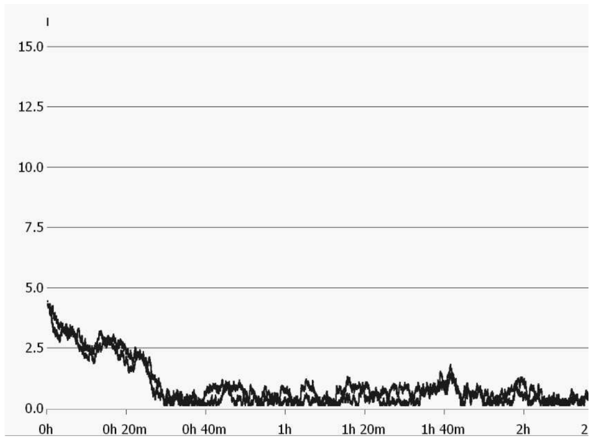
The primary goal of the system is to ensure that all incoming sheetsides are rasterized and available by their deadline for display. To assess whether a sheetside is available by its deadline, we defined a new measure known as "bitmap lifetime". Bitmap lifetime is measured as the time difference between when the rasterized image is made available in some 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.

Figure 1 presents the results of the simulation study in terms of bitmap lifetime. The plots show the bitmap lifetime values for each bitmap consumed by the system for both the MRCT and round-robin heuristics. The random assignment heuristic results are not plotted in Figure 1, because nearly all of the processed sheetsides had bitmap lifetimes that were nearly 0, i.e., random assignment continually failed to deliver bitmaps when they were needed. If a bitmap's lifetime value is close to 0, then the display device may 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 stopped to wait for bitmaps to become available. Attempting to abruptly stop the paper may ruin the result, e.g., by tearing the paper.

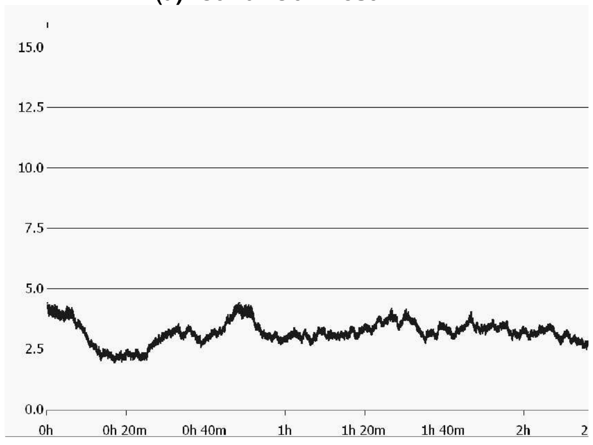
Because a bitmap must always be transferred to a display device, the minimum possible bitmap lifetime is t_{tran}^{bitmap}, i.e., near 0. Intuitively, if the bitmap lifetime for any bitmap has reached this value, then the bitmap was consumed by the display device as soon as it was created. In general, this implies that the display was ready to display the bitmap prior to it having been created. Alternatively, if the display is forced to wait for a bitmap to be created, then based on our model, this implies that the bitmap lifetime was t_{tran}^{bitmap}.

In the plots of Figure 1, the initial 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. In the case of the random assignment heuristic, the delays in processing, e.g., the total time that the printer was idle waiting for data, caused the simulation run time to be significantly greater than the other two heuristics. The round-robin heuristic is able to complete the entire run in only slightly more time than MRCT, however, it did experience a significant number of service interruptions as a result of its allocation decisions. In contrast, the MRCT heuristic is able to complete the entire run with no interruptions. For the MRCT heuristic, bitmap lifetimes were in the range of [1.64s,16s] throughout the simulation. These results demonstrate the utility of the mathematical model within the context of a resource allocation heuristic.



(a) round-robin result



(b) MRCT result

Fig. 1. Sample plots of the results for two of the three heuristics (a) round-robin, (b) MRCT.

VII. Related Work

The system studied in this research can be considered as a special-purpose distributed computing system, where workstations are dedicated to serving a single continuous stream of tasks (sheetsides) with hard deadlines and identical execution priorities. In contrast, a general-purpose computer cluster usually executes job requests from multiple users and jobs may have relative priorities. For example, batch job requests submitted to the NASA Ames iPSC/860 cluster controlled by the Portable Batch System (PBS), described in [3], are placed in one of the node's execution queues according to the priority level associated with a job. Jobs in each queue are fetched for execution in a first-come first-served manner with non-preemptive switching among the queues. The PBS Task Manager simply follows the mapping policy (manually created and modified on a day-to-day basis by a system administrator) while taking into consideration current system load.

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. In contrast, static mapping techniques are based on the assumption that the complete set of tasks considered for mapping is known *a priori*, i.e., the mapping is done prior to the execution of any of the tasks.

The general problem of dynamically allocating a class of independent tasks onto heterogeneous computing systems was studied in [14]. The primary objective in [14] 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 rasterizing every sheetside in a given job before the sheetside's deadline. Our MRCT heuristic attempts to map each sheetside to its estimated minimum RIP completion time workstation, which is analogous to the MCT heuristic of [14] attempting to map each task to its minimum completion time machine. However, the method of computing a completion time in [14] 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 MRCT approach, RIP completion time estimates for a task are continuously updated with the most current information regarding actual task completion times.

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.

VIII. Conclusion

The goal of this research was to rasterize dynamically arriving sheetsides (i.e., execute tasks) before a deadline to prevent an interruption of service. We presented a mathematical model of the distributed system operating in a dynamic environment where task execution times are uncertain. The mathematical model was used in the design of a resource management heuristic that clearly outperformed two commonly used approaches.

References

[1] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, Jul. 2004.

[2] I. Banicescu and V. Velusamy, "Performance of scheduling scientific applications with adaptive weighted factoring," in *Proceedings of the* 10^{th} *IEEE Heterogeneous Computing Workshop (HCW 2001),* 15^{th} International Parallel and Distributed Processing Symposium (IPDPS 2001), Apr. 2001.

[3] A. Bayucan, R. L. Henderson, C. Lesiak, B. Mann, T. Proett, and D. Tweten, "Portable batch system: External reference specification," NASA, Ames Research Center, Moffet Field, CA, Tech. Rep., 1996.

[4] E. G. Coffman, Ed., *Computer and Job-Shop Scheduling Theory*. New York, NY: John Wiley & Sons, 1976.

[5] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, Nov. 1989.

[6] I. Foster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann, 1999.

[7] A. Ghafoor and J. Yang, "A distributed heterogeneous supercomputing management system," *IEEE Computer*, vol. 26, no. 6, pp. 78–86, Jun. 1993.

[8] N. Gharachorloo, S. Gupta, R. F. Sproull, and I. E. Sutherland, "A characterization of ten rasterization techniques," in *SIGGRAPH '89: Proceedings of the* 16^{th} *Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM Press, 1989, pp. 355–368.

[9] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," *Journal of the ACM*, vol. 24, no. 2, pp. 280–289, Apr. 1977.

[10] H. A. James, K. A. Hawik, and P. D. Coddington, "Scheduling independent tasks on metacomputing systems," in *Proceedings of the* 12^{th} *International Conference on Parallel and Distributed Computing Systems (PDCS 99)*, Aug. 1999, pp. 47–52.

[11] D. L. Janovy, J. Smith, H. J. Siegel, and A. A. Maciejewski, "Models and heuristics for robust resource allocation in parallel and distributed computing systems," in *Proceedings of the* 21^{st} *International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Mar. 2007.

[12] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, vol. 6, no. 3, pp. 42–51, Jul. 1998.

[13] C. Leangsuksun, J. Potter, and S. Scott, "Dynamic task mapping algorithms for a distributed heterogeneous computing environment," in *Proceedings of the* 4^{th} *IEEE Heterogeneous Computing Workshop (HCW '95)*, Apr. 1995, pp. 30–34.

[14] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 107–121, Nov. 1999.

[15] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye, "Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness," *Journal of Supercomputing, Special Issue on Grid Technology*, accepted, to appear.

[16] (2007) Opnet technologies, inc. Accessed Feb. 20, 2007. [Online]. Available: http://www.opnet.com/

[17] V. Shestak, J. Smith, H. J. Siegel, and A. A. Maciejewski, "Iterative algorithms for stochastically robust static resource allocation in periodic sensor driven clusters," in *Proceedings of the* 18^{th} *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2006)*, Nov. 2006, pp. 166–174.

[18] ——, "A stochastic approach to measuring the robustness of resource allocations in distributed systems," in *Proceedings of the 2006 International Conference on Patallel Processing (ICPP 2006)*, Aug. 2006, pp. 6–12.

[19] V. Shestak, J. Smith, R. Umland, J. Hale, P. Moranville, A. A. Maciejewski, and H. J. Siegel, "Greedy approaches to static stochastic robust resource allocation for periodic sensor driven systems," in *Proceedings of The 2006 International Conference on Parallel and Distributed Processing Techniques and Applications*, vol. 1, Jun. 2006, pp. 3–9.

[20] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," in *Proceedings of the* 5^{th} *IEEE Heterogeneous Computing Workshop (HCW '96)*, 1996, pp. 86–97.

[21] J. Smith, L. D. Briceño, A. A. Maciejewski, and H. J. Siegel, "Measuring the robustness of resource allocations in a stochastic dynamic environment," in *Proceedings of the* 21^{st} *International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Mar. 2007.

[22] X. Tang and 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)*, Aug. 2000, p. 373.

[23] M. Wu and W. Shu, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems," in *Proceedings of the* 9^{th} *IEEE Heterogeneous Computing Workshop*, Mar. 2000, pp. 375–385.

[24] D. Xu, K. Nahrstedt, and D. Wichadakul, "Qos and contention-aware multi-resource reservation," *Cluster Computing*, vol. 4, no. 2, pp. 95–107, Apr. 2001.