PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OTIMIZANDO O ESCALONAMENTO DE *JOBS* NO PROCESSO DE RASTERIZAÇÃO DE DOCUMENTOS PERSONALIZÁVEIS

CAROLINA MARQUES FONSECA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Luiz Gustavo Leão Fernandes

Porto Alegre
2011

F676o Fonseca, Carolina Marques
Otimizando o escalonamento de jobs no processo de rasterização de documentos personalizáveis / Carolina Marques Fonseca. – Porto Alegre, 2011.
74 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Luiz Gustavo Leão Fernandes

1. Informática. 2. Documentos – Personalização. 3. Impressão Digital. I. Fernandes, Luiz Gustavo Leão. II. Título

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Otimizando o Escalonamento de Jobs no Processo de Rasterização de Documentos Personalizáveis**", apresentada por Carolina Marques Fonseca, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 29/03/2011 pela Comissão Examinadora:

Prof. Dr. Luiz Gustavo Leão Fernandes– Orientador — PPGCC/PUCRS

Prof. Dr. Fabiano Passuelo Hessel - — PPGCC/PUCRS

Prof. Dr. Marilton Sanchotene de Aguiar – — UFPEL

Homologada em 02/04/2013, conforme Ata No. 005 pela Comissão Coordenadora.

Prof. Dr. Fernando Luís Dotti
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

Dedico este trabalho à minha família, amigos, colegas e orientador pelo apoio, força, incentivo e amizade. Sem eles nada disso seria possível.

AGRADECIMENTOS

À minha família, principalmente minha mãe por sempre me apoiar nas minhas decisões, pelo amor, carinho e dedicação. À meus irmãos, que mesmo longe sempre estiveram ao meu lado me incentivando. Amo muito vocês!

Ao meu orientador Luiz Gustavo Fernandes, pela oportunidade, conselhos e amizade. Pela orientação e por todo conhecimento repassado, eu agradeço.

Ao LAD (Laboratório de Alto Desempenho) da PUCRS, por disponibilizar a estrutura para a execução de todos os testes, e ao colega Antonio de Lima, pelo apoio na realização dos testes.

Aos amigos e colegas do GMAP (Grupo de Modelagem de Aplicações Paralelas), que de alguma forma tentaram me ajudar, apoiar, e fizeram com que esses dois anos de mestrado passassem de forma descontraída e alegre. Em especial agradeço aos amigos Rafael Nemetz, Thiago Nunes e Mateus Raeder que sempre estiveram dispostos a me ajudar, pela amizade e pelas conversas. Obrigada a todos!

Por fim, agradeço a todas as pessoas que de alguma forma contribuíram para meu desenvolvimento profissional e pessoal.

OTIMIZANDO O ESCALONAMENTO DE *JOBS* NO PROCESSO DE RASTERIZAÇÃO DE DOCUMENTOS PERSONALIZÁVEIS

RESUMO

Impressoras digitais têm melhorado constantemente a sua velocidade nos últimos dez anos. Enquanto isso, cresceu a necessidade de personalização e customização de documentos. Como consequência disto, o processo de rasterização tornou-se uma etapa que demanda um grande poder computacional. As empresas especializadas na publicação digital já estão utilizando múltiplos RIPs (responsáveis por realizar a rasterização dos documentos) e estratégias de paralelização para acelerar o processo de rasterização que é realizado sobre cada página do documento. No entanto, estas estratégias não são otimizadas para garantir a melhor utilização dos recursos dos RIPs. Dependendo das características dos documentos, o processo de rasterização pode ser mais demorado, criando assim um gargalo indesejado. Em trabalhos anteriores, algumas estratégias foram introduzidas para aumentar o desempenho deste procedimento através do uso de técnicas de processamento paralelo e distribuído. Apesar dos bons resultados obtidos, detectou-se que algumas otimizações poderiam ser propostas com o intuito de melhorar o balanceamento de carga entre as unidades de rasterização. Nesse cenário, para maximizar a utilização dos recursos, novas estratégias foram propostas com o intuito de distribuir de maneira balanceada a carga de trabalho entre as unidades de rasterização, levando em conta novas características dos documentos, como transparência e reusabilidade de objetos. Os resultados obtidos confirmam que era possível melhorar o desempenho através da exploração dessas novas características.

Palavras-chave: PDF; Rasterização; Balanceamento; Transparência; Reusabilidade.

OPTIMIZING JOBS SCHEDULING IN THE PERSONALIZED DOCUMENTS RIPPING PROCESS

ABSTRACT

Digital presses have consistently improved their speed in the past ten years. Meanwhile, the need for document personalization and customization has increased. As a consequence of these two facts, the traditional RIP (Raster Image Processing) process has become a highly demanding computational step in the print workflow. Print Service Providers (PSPs) are now using multiple RIP engines and parallelization strategies to speed up the whole ripping process which is currently performed on a per-page basis. Nevertheless, these strategies are not optimized in terms of ensuring the best resorces utilization for the RIP engines. Depending on the input document jobs characteristics, the ripping step may not achieve the print-engine speed creating a unwanted bottleneck. In previous works, some strategies have been introduced to increase the performance of the ripping procedure by using techniques of parallel and distributed computing. Despite of the good results achieved, it was possible to identify that some optimizations could be proposed in order to improve the load balance between ripping units. In this scenario, to maximize the use of resources, new strategies have been proposed to distribute the workload balancing between ripping units considering new characteristics of documents such as transparency and reusability of objects. The obtained results confirm that it was indeed possible to improve the performance through the use of these new features.

Keywords: PDF; Ripping; Balancing; Transparency; Reusability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

COW	*Cluster Of Workstations*
DPI	*Dots Per Inch*
FFD	*First Fit Decreasing*
GB	*Gigabytes*
GHz	*Gigahertz*
HP	*Hewlett Packard*
JPEG	*Joint Photografics Experts Group*
LS	*List Scheduling*
LPT	*Largest Processing Time first*
MPI	*Message Passing Interface*
PDF	*Portable Document Format*
PDL	*Page Description Language*
PPML	*Personalized Print Markup Language*
PS	*PostScript*
PSP	*Print Service Provider*
RAM	*Random Access Memory*
RIP	*Raster Image Processor*
RIPping	*Raster Image Processing*
TB	*Terabytes*
VDP	*Variable Data Printing*
XML	*Extensible Markup Language*
XObjects	*eXternal Objects*

SUMÁRIO

1. INTRODUÇÃO

Com o surgimento das impressoras digitais com alta vazão (com o poder de imprimir grande quantidade de páginas em pouquíssimo tempo), procedimentos automatizados para a criação e transformação de documentos se tornaram necessários afim de suprir a nova demanda. Uma nova disciplina, *Variable Data Printing* (VDP) [26], que provê diversas técnicas, padrões, conceitos e tecnologias foi introduzida, permitindo a personalização de documentos. Com isto, foram criadas linguagens para a descrição de documentos personalizados e processos para a impressão correta destes. Estas linguagens definem partes estáticas e dinâmicas para os documentos criados. Assim, um mesmo *layout* pode ser aplicado para diferentes instâncias de documentos. Exemplos desses tipos de documentos são os formatos PDF (*Portable Document Format*) [2] e o PS (*PostScript*) [1].

Atualmente, a maioria das impressoras digitais não é capaz de interpretar as linguagens que apresentam flexibilidade (ou seja, que permitem a definição de áreas estáticas e dinâmicas de um documento). Assim, tornam-se necessários procedimentos adicionais que possibilitem a impressão dos documentos de forma correta. Dois destes procedimentos são a renderização e a rasterização de documentos. Estes processos têm como objetivo fornecer ao dispositivo de impressão o documento no formato necessário (formato *bitmap* por exemplo) para que a impressão seja realizada de maneira correta.

Com a introdução da personalização de documentos, as empresas especializadas na publicação digital (*Print Service Providers* - PSPs) necessitam realizar estes procedimentos sobre cada página dos documentos personalizados, o que acaba provocando um alto custo computacional para o processamento.

1.1 Motivação

O formato PDF é amplamente utilizado no âmbito da impressão digital por apresentar diversas características vantajosas. Ele é obtido através do processo de renderização sobre as porções variáveis de um documento (descritas por uma linguagem de formatação) e logo após é feito o processo de rasterização possibilitando sua impressão. Porém, as etapas de renderização e rasterização são muito custosas e normalmente centralizadas em uma única unidade de processamento, acabando por representar um gargalo no processo de impressão, no caso da existência de milhares de instâncias de documentos a cada *job*.

Como as PSPs possuem uma fila de *jobs* com diversos conjuntos de documentos a serem impressos, elas normalmente utilizam impressoras com um alto poder de processamento. Assim, todas as fases da preparação de um documento devem ser finalizadas em um tempo limitado para não ocorrer a subutilização das impressoras disponíveis.

Geralmente, as PSPs utilizam as impressoras em paralelo para poder atender um número maior de requisições de impressão de pré-processamento de um documento deve aumentar proporcionalmente

para mantê-las trabalhando.

Algumas estratégias na etapa de renderização foram tratadas em trabalhos passados através da utilização de técnicas de alto desempenho para acelerar o processamento [23]. Com relação a etapa de rasterização, algumas pesquisas começaram a ser desenvolvidas recentemente [22], com o objetivo de estabelecer mecanismos que permitam a reorganização da fila de impressão de *jobs*. Com estes mecanismos, foi possível aumentar a vazão de fase de pré-processamento. No entanto, alguns aspectos desse processo de reorganização ainda podem ser melhor explorados.

1.2 Objetivos

O objetivo deste trabalho trata-se do desenvolvimento de estratégias a partir de diferentes cenários, para obter um bom balanceamento de carga em ambientes de impressão distribuídos durante a fase de rasterização de documentos PDF. As estratégias propostas levam em consideração diferentes características presentes em documentos PDF como a transparência e a reusabilidade de imagens.

Este trabalho também tem como objetivo desenvolver um escalonador chamado Roteador Adaptativo de *Jobs* (RAJ), o qual criará um arquivo XML (*eXtensible Markup Language*) [7] contendo informações sobre o documento PDF que servirá de base para determinar quais das estratégias desenvolvidas será utilizada em cada documento. Para a validação das estratégias e do escalonador RAJ, um conjunto de documentos PDF com diferentes características foi criado.

1.3 Estrutura do Trabalho

Este volume está dividido em nove capítulos, sendo o primeiro deles a presente introdução. O restante está organizado da seguinte forma:

- **Capítulo 2**: apresenta algumas técnicas ligadas ao processo de impressão, explicitando os passos e as fases existentes desde a criação de um documento personalizável até a sua impressão. Também são apresentadas neste capítulo as estratégias base existentes no âmbito da rasterização de documentos. Além disso, é descrito um estudo inicial sobre o formato PDF, abrangendo suas características e sua composição. Por fim, são apresentados alguns trabalhos relacionados;

- **Capítulo 3**: apresenta as classificações, algoritmos e formalismos sobre escalonamento;

- **Capítulo 4**: descreve o escalonamento baseado em perfil de documentos PDF;

- **Capítulo 5**: apresenta a abordagem proposta deste trabalho, detalhando as novas estratégias e o escalonador RAJ;

- **Capítulo 6**: detalha todo o ambiente de teste, assim como seus casos de testes, e a avaliação das estratégias;

- **Capítulo 7**: apresenta o ambiente de teste e os casos de testes, assim como a validação do escalonador RAJ;
- **Capítulo 8**: conclui o trabalho realizado e apresenta possíves trabalhos futuros a serem realizados.

2. CENÁRIO E CONTEXTUALIZAÇÃO

Este capítulo apresenta algumas técnicas envolvidas no processo de impressão, relatando os passos e fases existentes desde a criação de um documento personalizável até a sua impressão. Além disso, são apresentadas as estratégias base existentes em ambientes de rasterização tradicional, mostrando suas principais vantagens e desvantagens. A composição do formato PDF é abordada em seguida, mostrando algumas de suas principais características.

2.1 O Processo de Impressão

A personalização de documentos é uma nova tendência criada devido à grande eficiência na impressão digital de documentos. Com isto, foram desenvolvidas linguagens para a descrição de documentos personalizados [26] e processos para a impressão correta destes no formato de documentos PDF, por exemplo.

Para personalizar documentos, estas linguagens definem partes estáticas e dinâmicas para os documentos criados. O *designer* de um projeto é o responsável por estabelecer um *template* de um documento contendo estas partes estáticas e dinâmicas. Este *templante*, então, será combinado com registros do banco de dados que contêm informações diferentes, as quais serão usadas para completar cada documento, gerando assim diferentes instâncias dos documentos personalizados.

Atualmente, a maioria das impressoras digitais não são capazes de interpretar as linguagens que apresentam flexibilidade. Assim, tornam-se necessários procedimentos adicionais que possibilitem a impressão dos documentos corretamente.

Dois destes procedimentos são as previamente citadas renderização e rasterização de documentos. Ambos apresentam um alto custo computacional e, se não otimizados, representam um gargalo no processo de impressão de documentos em impressoras de alta vazão.

A etapa de renderização refere-se ao processo de interpretação de uma determinada linguagem de formatação não usual a um leitor humano, como, PPML (*Personalized Print Markup Language*) [6], por exemplo. Através da aplicação desta etapa, tem-se um documento descrito por uma linguagem de alto nível de abstração (por exemplo, PDF), tornando possível sua visualização. Na etapa de renderização, alguns trabalhos envolvendo processamento paralelo já foram realizados para otimizar o desempenho [21] [20] [23]. Já na etapa de rasterização, trabalhos mais recentes vêm sendo realizados [19]. O foco do presente estudo é esta etapa, que é descrita em maiores detalhes a seguir.

A etapa de rasterização ou *RIPping* (*Raster Image Processing*), consiste na conversão do documento em um formato conhecido pelas impressoras, uma vez que elas não são capazes de interpretar as linguagens de alto nível como PS e PDF para descrição de documentos. Esta conversão é feita geralmente para o formato *bitmap* (matriz de pontos com suas correspondentes cores, que indicam corretamente ao dispositivo de impressão em questão como apresentar o conteúdo descrito no documento).

As RIP *engines* (ou simplesmente RIPs) são responsáveis por realizar a rasterização dos documentos (rasterizadores). A Figura 2.1 ilustra o processo descrito acima.

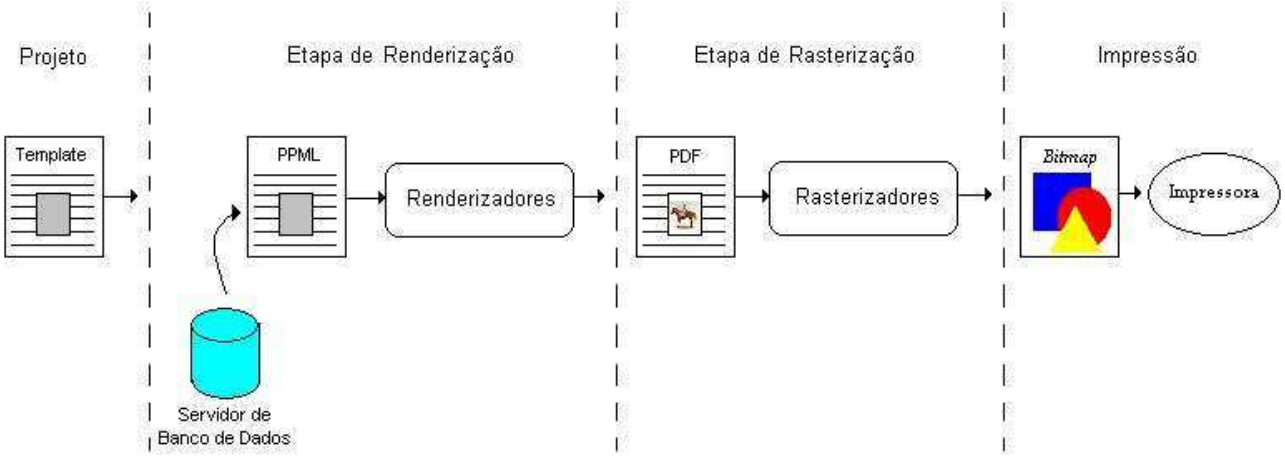

Figura 2.1: *Workflow* do processo de impressão de documentos personalizáveis.

Através do uso de técnicas relacionadas ao processamento paralelo e distribuído, algumas estratégias já foram aplicadas para aumentar o desempenho desta etapa [22]. Entretanto, tais estratégias apresentam alguns problemas, como a impossibilidade de garantir um balanceamento de carga justo para quaisquer sequências de *jobs* contendos documentos personalizados, por exemplo.

2.2 Estratégias de Rasterização Base

Em um ambiente de rasterização tradicional, as estratégias existentes baseiam-se em sistemas paralelos e distribuídos para aumentar a vazão e o desempenho de tal fase. Assim, vários RIPs são aplicados em conjunto para rasterizar uma certa fila de *jobs* de forma paralela. Com isto, através da análise das estratégias é possível verificar as vantagens e desvantagens existentes. São identificadas três estratégias que são aplicadas para acelerar a rasterização dos *jobs*:

1. **Alocar um RIP por *job* (Figura 2.2)**: cada RIP existente irá processar um *job* inteiro. Quando um RIP acabar de processar o *job*, ele requisitará mais trabalho para o escalonador. Assim, existem duas situações que podem acontecer: a distribuição da carga é injusta (tamanho dos *jobs* podem variar muito) ou vários RIPs ficam ociosos (quando o número de *jobs* for menor do que o número de RIPs disponíveis). Este tipo de cenário funciona bem para *jobs* pequenos, com alta reusabilidade pois como todas as imagens ficam no mesmo RIP seu tempo de processamento acaba diminuindo porque a imagem não precisa ser rasterizada novamente;

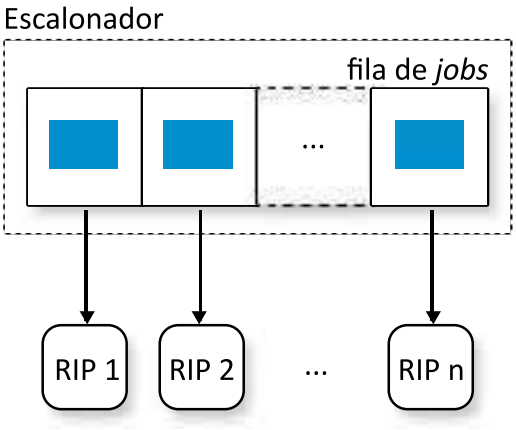

Figura 2.2: Alocar um RIP por *job*. Extraído de [22] .

2. **Alocar todos RIPs para um único *job* (Figura 2.3)**: todos os RIPS estarão disponíveis para um único *job*. Esta é a abordagem "força-bruta", na qual cada RIP irá processar uma porção de um dado *job* (porção composta por no mínimo uma página). O escalonador é o responsável por definir as porções, passando apenas as páginas de cada fragmento para os RIPs, para então os RIPs realizarem as quebras dos *jobs*. Esta estratégia funciona bem com *jobs* que apresentam baixa reusabilidade (as imagens iguais ficarão em RIPS diferentes) e demandam alto poder computacional;

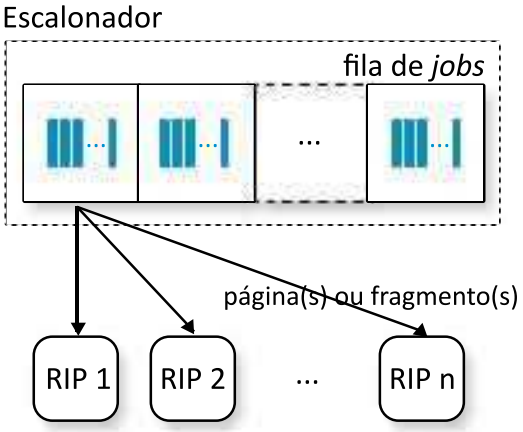

Figura 2.3: Alocar todos RIP para um único *job*. Extraído de [22] .

3. **Alocar um número fixo de RIPs por *job* (Figura 2.4)**: esta configuração é baseada no fato de que os *jobs* padrões das PSPs necessitam de um determinado número de recursos para manterem as impressoras continuamente trabalhando. Uma desvantagem é o fato de que um conjunto de RIPs somente será alocado para um novo *job* assim que todos os RIPs estiverem livres, ou seja, no momento em que o *job* atual for processado por completo.

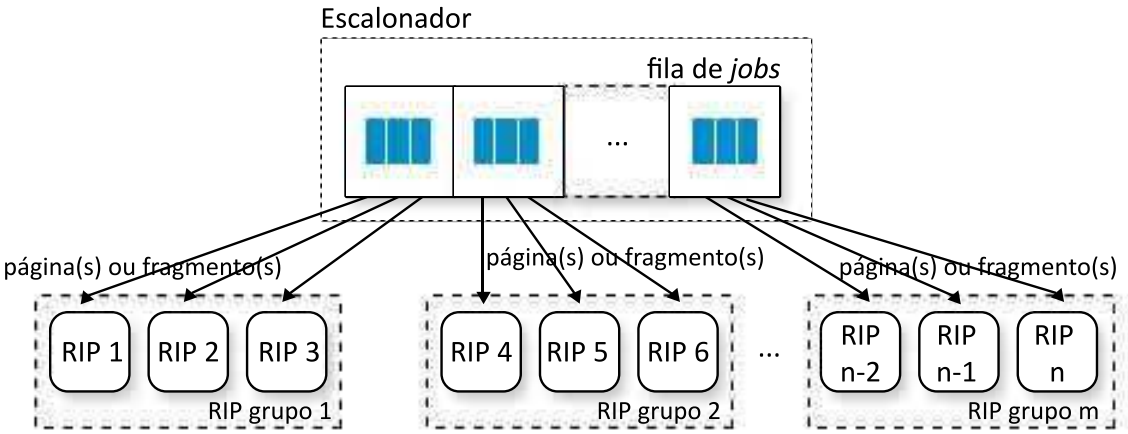

Figura 2.4: Alocar um número fixo de RIPs por *job*. Extraído de [22] .

Nenhuma das estratégias discutidas acima, garante o melhor balanceamento de carga entre os RIPs. Porém, a segunda e a terceira apresentam uma eficiência maior do que a primeira estratégia.

Na estratégia 1 supõe-se como exemplo, 3 RIPs e 3 *jobs*. O primeiro *job* (contendo 10 páginas) é executado em 100s, o segundo (contendo 3 páginas) em 30s e o terceiro (contendo 5 páginas) em 50s. Com estes valores, nota-se que o segundo RIP ficará ocioso por um tempo muito grande (70s).

Já nas estratégias 2 e 3 como os *jobs* são quebrados em grãos (páginas ou grupo de páginas) o balanceamento de carga entre os RIPs acaba sendo melhor. Supõe-se o mesmo exemplo citado acima para estas duas estratégias. Cada página será distribuída de forma que cada RIP fique com uma quantidade similar ao dos outros RIPs. Em um primeiro momento, as 10 páginas do primeiro *job* são distribuídas. Assim, o primeiro RIP ficaria com 4 páginas o segundo e o terceiro com 3. Os RIPs 1 e 2 ficariam ociosos por 10 segundos até que o primeiro *job* fosse rasterizado. Quando todas as páginas forem rasterizadas, o próximo *job* da fila começa a ser distribuído (*job* com 3 páginas). Com isto, o primeiro, o segundo e o terceiro RIP ficam com 1 página cada um. Após este *job* ser rasterizado, o último *job* da fila começa a ser processado (*job* com 5 páginas). Assim, o primeiro e o segundo RIP ficam com 2 páginas e o terceiro com 1 página (fazendo com que o RIP 3 ficasse 10 segundos ocioso). Portanto, nota-se que com estas duas estratégias os RIPs não ficam tão ociosos (20s) quanto na primeira estratégia.

2.3 Informações Presentes em Documentos PDF

O formato PDF é altamente difundido entre as PSPs para a descrição de documentos. Este formato é uma PDL (*Page Description Language*) utilizada para descrever gráficos de um documento e provê uma representação que independe de *software*, *hardware* ou sistema operacional para criá-lo. Algumas das características e vantagens fornecidas pelo formato PDF são:

- **Portabilidade**: documento PDF é portável para qualquer plataforma, pois é armazenado como um arquivo binário ao invés de ser representado em texto puro;
- **Compressão**: o documento PDF suporta padrões atuais de compressão como o JPEG (*Joint Photografics Experts Group*) [24];
- **Segurança**: um documento PDF pode ser criptografado e descriptografado por diversos meios;
- **Gerenciamento de Fontes**: é possível adicionar ao documento PDF, diversos tipos de fontes com diferentes formatos.

Algumas destas vantagens já existiam no formato PS (no qual o PDF é baseado), herdando não somente as características e vantagens citadas anteriormente, mas também outros atributos, como o fato de um documento ser formado por uma série de objetos, por exemplo. Estes objetos, são os responsáveis pela descrição da aparência das páginas. Eles são dispostos de maneira sequencial, mas não são necessariamente lidos ou interpretados desta maneira.

Na especificação do formato PDF são incluídos oito tipos básicos de objetos [2]: *Strings*, Valores Booleanos, Valores Numéricos, Nomes, Vetores, Dicionários de Dados, *Stream* Objetos e Objeto Nulo. Estes objetos podem ser utilizados no escopo do documento, auxiliando a descrição do conteúdo de forma mais flexível.

Além disso, o formato PDF utiliza gráficos para representar o conteúdo de uma ou mais páginas. Os gráficos podem ser divididos em grupos específicos conforme sua funcionalidade no documento PDF, como, por exemplo, o grupo dos textos (tratado como grupo especial de gráficos pelo PDF, pela sua grande utilização na criação de documentos). Os gráficos podem ser classificados em 5 tipos:

- ***Path Objects***: representam formas arbitrárias, trajetórias, regiões e *paths*;
- ***eXternal Objects***: representam elementos que podem ser reusáveis. Três sub-tipos de *xObjects* são conhecidos:

  - *Images xObjects*: descrevem imagens *bitmap* que representam seu conteúdo através de uma matriz de *pixels*;

  - *Postscript xObject*: objetos que definem seu conteúdo através de PS *commands*. Estes objetos estão deixando de ser usados em PDFs;

  - *Groups (forms) xObjects*: grupo de objetos gráficos, usados para definir propriedades comuns dentro de um grupo.

- ***Inline Image Objects***: definem uma imagem diretamente dentro do PDF (não pode ser reusável). Esses objetos contêm diversas limitações, sendo a mais relevante o tamanho da imagem;

- ***Shading Pattern Objects***: descrevem uma forma geométrica (que tem sua cor definida por uma função arbitrária);
- ***Text Objects***: descrevem as porções de texto do documento, incluindo o formato e suas características, tais como fonte, tamanho, espaçamento entre letras etc.

Estes objetos possuem características importantes, como a transparência e a reusabilidade, que influenciam no balanceamento de carga dos RIPs [22]. A **transparência** de imagens é a propriedade que permite que se visualize de forma sobreposta duas ou mais imagens que ocupam regiões iguais ou o fundo do documento através da imagem. Já a **reusabilidade** ocorre quando instâncias da mesma imagem são utilizadas mais de uma vez na mesma página ou em páginas diferentes.

2.4 Trabalhos Relacionados

Em trabalhos passados [20] [21] [23], algumas estratégias para elevar a vazão do processo de renderização foram apresentadas. Já para o processo de rasterização, foi desenvolvida uma ferramenta chamada PDF *Profiler* [19], com a capacidade de analisar e fornencer todas as informações contidas em um documento PDF. Com o uso desta ferramenta, foi possível realizar a análise do perfil dos *jobs*, contendo informações sobre os objetos que os compõem. A análise realizada concentrou-se no formato PDF, por ser este um formato difundido na descrição de documentos personalizados.

Para a execução do PDF *Profiler*, deve ser fornecido um documento XML que define as informações que serão analisadas, juntamente com o documento PDF do qual se deseja extrair as informações. Após esta execução, a ferramenta gera um documento XML de saída com os resultados obtidos.

Para cada página do documento PDF, as seguintes informações são fornecidas: área total da página, textos, área total das imagens com transparência e com ou sem reusabilidade, se contém escopo de *image XObjects* e se contém elementos com transparência. A partir das informações fornecidas pelo PDF *Profiler*, busca-se encontrar algumas estratégias para obter um balanceamento de carga mais justo para cada RIP.

3. ESCALONAMENTO

Um dos maiores desafios da área de sistemas computacionais paralelos e distribuídos é o desenvolvimento de técnicas eficientes para a distribuição de processos de aplicações paralelas entre os elementos de processamento. Esta distribuição de processos é conhecida como escalonamento de processos (*job scheduling*) e tem como objetivo minimizar o tempo médio de resposta e/ou melhorar a utilização de recursos.

Para melhorar o desempenho de sistemas computacionais paralelos e distribuídos, algumas características devem ser levadas em consideração, como o *hardware* da rede de comunicação, o sistema operacional, a presença de aplicações com diferentes características e a escolha do melhor tamanho de tarefa (grão) a ser transmitido e/ou processado [16].

Esta última característica é uma das primeiras preocupações que se deve considerar, pois caso se defina um tamanho de grão ruim, o *overhead* existente para a transmissão pode não compensar o ganho de desempenho obtido através da divisão de trabalho. Assim, para que a escolha do grão seja boa deve-se basear nas características específicas de cada aplicação.

Logo após a tarefa ser definida, deve-se estabelecer a melhor estratégia de distribuição destas tarefas entre os processos para que não ocorra uma sobrecarga ou subcarga. Se a escolha desta estratégia não for boa, a eficiência pode ficar comprometida. Na Figura 3.1 esta situação pode ser notada.

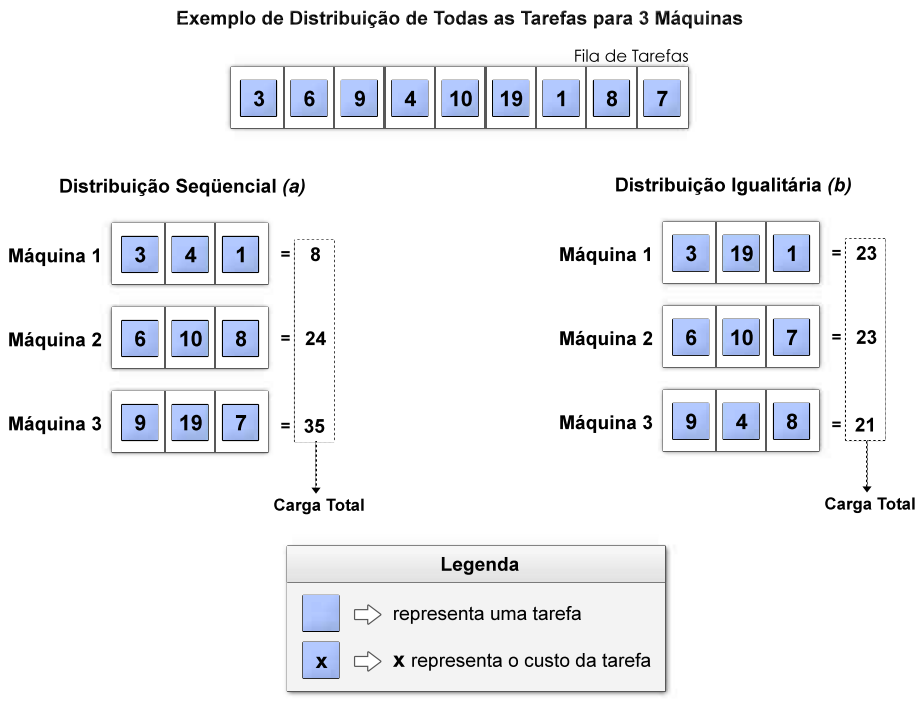

Figura 3.1: Influência das estratégias de escalonamento. Extraído de [22]

Na distribuição sequencial, na qual se associa as tarefas uma a uma para cada máquina (primeira tarefa para a máquina 1, segunda tarefa para a máquina 2, terceira tarefa para a máquina 3, quarta tarefa para a máquina 1 e assim por diante), tem-se uma sobrecarga na máquina 3 de 35. Isto ocorre porque o processamento de toda a fila só acaba quando todas as máquinas já terminaram suas tarefas. Assim, o tempo de processamento é igual ao tempo em que a última máquina acabar de processar a tarefa.

Já com uma distribuição balanceada esta situação de sobrecarga não irá acontecer, pois leva-se em consideração a carga da tarefa. Assim, o tempo de finalização da fila de tarefas será menor. A procura pela divisão balanceada de tarefas caracteriza o problema de escalonamento [29].

Em [10] foi introduzido uma notação para se agrupar os problemas de escalonamento com características parecidas, afim de analisar os aspectos que influenciam cada grupo. Esta notação possui três tipos de características diferentes em relação ao problema de escalonamento.

A primeira característica refere-se ao ambiente de execução da estratégia de escalonamento (relacionado ao *hardware*). A segunda característica é em relação às pecuiliaridades de cada tarefa e a restrições da estratégia de escalonamento. Por fim, a terceira estratégia é em relação a medida de desempenho (tempo para terminar uma tarefa) que se deseja otimizar com a aplicação da estratégia de escalonamento. Nas seções a seguir será apresentada uma classificação para os problemas de escalonamento e alguns algoritmos de escalonamento existentes.

3.1 Classificação

Pode-se classificar o escalonamento em duas situações distintas [29], dependendo da disponibilidade de informações sobre as tarefas: determinístico e não determinístico.

3.1.1 Escalonamento Determinístico

O escalonamento determinístico é aquele em que todas as características das tarefas e as relações entre cada uma delas são conhecidas previamente à execução da aplicação. Dois tipos de algoritmos se destacam: algoritmos ótimos e sub-ótimos.

Algoritmos ótimos são os que se preocupam em encontrar a melhor solução para o problema de escalonamento. Entretanto, este tipo de algoritmo necessita de grande quantidade de informações. Além disso, sabe-se que em geral os problemas de escalonamento são NP-Completos [4] [25]. O caso ideal seria utilizar os algoritmos ótimos para todas as situações, já que se teria a utilização da melhor configuração. Porém, este tipo de algoritmo acaba gerando um alto custo em termos de desempenho.

Já os algoritmos sub-ótimos buscam uma solução através de técnicas específicas (como a heurística) sem nenhuma garantia que seja a melhor solução. Porém, sabe-se que a solução encontrada é próxima da ótima. Apesar disso, este algoritmo obtém resposta em tempo polinomial, o que não seria possível com o algoritmo ótimo. Assim, com a utilização do algoritmo sub-ótimo consegue-se um número maior de resoluções de problemas do que os algoritmos ótimos.

3.1.2 Escalonamento Não Determinístico

O escalonamento não determinístico é aquele em que não se tem todas as informações necessárias sobre as tarefas antes da execução do programa, provendo uma decisão durante a execução do programa. Com isto, acaba ocorrendo um *overhead* na própria aplicação. Assim, dois tipos de algoritmos se destacam: algoritmos estáticos e dinâmicos.

Algoritmos estáticos realizam uma análise (durante o tempo de execução do programa) sobre toda tarefa antes de distribuí-las, conseguindo assim a melhor maneira de realizar o escalonamento. Este algoritmo é mais vantajoso caso o processamento da análise for feito em um período de tempo que seja compensado pelo ganho de desempenho obtido através da estratégia de escalonamento estabelecida.

Os algoritmos dinâmicos propõem a intercalação da análise com a distribuição das tarefas. Assim, a análise é realizada apenas de um conjunto de tarefas, que após são distribuídas para serem processadas, partindo para a anáise de outro conjunto. Com isto, a partir de determinado momento o processo de escalonamento passa a ser concorrente ao cálculo do custo computacional de cada tarefa.

3.2 O Problema $Pm||C_{max}$

Um dos problemas existentes na área de escalonamento é denominado $Pm||C_{max}$ [17]. Este problema diz respeito ao escalonamento de um conjunto de n tarefas em m máquinas iguais (reduzindo o tempo para completar a última tarefa) e é considerado como NP-difícil [8], impossibilitando o desenvolvimento de algoritmos ótimos para resolver este problema. Assim, diversos algoritmos através da heurística foram desenvolvidos para obter um resultado o mais próximo do ótimo possível.

Para avaliar estes algoritmos no contexto de problemas NP-dificeis, um método que trata da distância do resultado do algoritmo, considerando o seu pior caso, em relação ao ótimo é utilizado. Este método é conhecido como análise de competitividade [27] [15].

3.2.1 Algoritmos de Escalonamento

Entre os diversos algoritmos para resolver o problema $Pm||C_{max}$ os mais difundidos são: *List Scheduling* (LS) [11], *Largest Processing Time First* (LPT) [10] e *Multifit* [5]. A seguir o funcionamento, vantagens e desvantagens são apresentadas.

List Scheduling

O algoritmo LS tornou-se a base para o desenvolvimento de muitos algoritmos de escalonamento. A partir dele também foi possível descobrir outros problemas na área escalonamento. O funcionamento deste algoritmo é da seguinte forma: dada uma fila A de tarefas organizadas de qualquer forma, sempre deve-se transmitir a primeira tarefa da fila para uma máquina ociosa, isto é, assim que ela acabar de processar a tarefa atual.

Considerando a análise de competitividade para m máquinas, o LS é $(2-\frac{1}{m})$-*competitive* [10], isto é, sua competitividade piora à medida que mais máquinas são utilizadas. Assim, seu pior caso é quando a última tarefa da fila tem o maior tempo de processamento.

Largest Processing Time First

O algoritmo LPT foi construído com o objetivo de melhorar a competitividade do algoritmo LS. Assim, focou-se na prevenção do pior caso do algoritmo LS (última tarefa da fila tem o maior tempo de processamento). Desta maneira, o LPT introduz a ordenção das tarefas da fila de maior tempo de processamento para o menor tempo. Após esta ordenação é feito o algoritmo LS normalmente. Desta maneira, ameniza-se o efeito de perda de desempenho provocada pelo não balanceamento de cargas grandes entre os processos.

Multifit

Logo após o LPT ser desenvolvido, o primeiro algoritmo a ter um ganho relativamente bom foi o algoritmo *Multifit*. Este algoritmo é baseado en técnicas *bin-packing* [14]. Esta técnica trata do empacotamento de n tarefas, com uma carga qualquer e com um número finito de *bins* (pacotes), onde o número de *bins* utilizados deve ser o menor possível. Como o problema do *bin-packing* é NP-completo [3] diversas heurísticas foram desenvolvidas para resolver este problema. Para este algoritmo *Multifit* a heurística utilizada é o *First-Fit Decreasing* (FFD) a qual agrupa as tarefas em até m (número de máquinas) *bins*. A estratégia FFD funciona da seguinte maneira:

1. organiza-se as tarefas conforme seus tempos de processamento de forma decrescente em uma sequencia;
2. cada tarefa da sequencia é adicionada no *bin* de forma que a capacidade do *bin* não ultrapasse um limite pré-definido;
3. os passos 1 e 2 são executados até que todas as tarefas etsejam em um *bin*.

4. ESCALONAMENTO BASEADO EM PERFIL DE DOCUMENTOS PDF

No Capítulo 2 (Seção 2.2), as três estratégias de rasterização base apresentadas (com o objetivo de acelerar a rasterização dos *jobs*) utilizam apenas o algoritmo LS para realizar o escalonamento de tarefas. Este algoritmo foi um dos primeiros estabelecidos na área de escalonamento, não possuindo a melhor competitividade dentre os algoritmos existentes. Isto pode ocasionar uma má utilização dos recursos, prejudicando o desempenho que poderia ser obtido.

Com o intuito de melhorar a situação existente, uma abordagem foi proposta em [19]. Esta abordagem foi feita através do desenvolvimento de um escalonador, o qual aplica dois algoritmos de escalonamento que levam em conta o custo computacional das tarefas, na tentativa de obter um balanceamento de carga mais justo. Estes algoritmos são o LPT e o *Multifit*. A seguir, são apresentados os passos necessários para que esta abordagem fosse desenvolvida.

4.1 Métricas

Foram analisadas diversas métricas para estimar o custo computacional do processo de rasterização de um documento PDF. A partir do conhecimento do funcionamento dos RIPs e das características principais dos *jobs* das PSPs, foi possível decidir quais objetos e características deveriam ser levados em consideração. Assim, os objetos gráficos (textos e imagens) em conjunto com as páginas foram utilizados para compor as métricas para avaliar o custo computacional de um documento PDF.

Para validar estas métricas, foram realizados alguns experimentos [19]. Através destes foi possível estabelecer uma maneira para calcular o custo computacional associado a cada uma das métricas avaliadas (textos, imagens e páginas).

Para o cálculo do custo de páginas em branco, observou-se o incremento do fator de relevância, que aumenta quanto maior for o número de páginas ou as dimensões da página. Isto se refere ao número de operações de E/S que deverá ser realizada. A partir disto, foi definido o custo de uma página.

No cálculo do custo dos textos, em um primeiro momento foi considerado o impacto do aumento da quantidade de texto e do tamanho da fonte em um documento PDF. Porém, com os testes realizados em [19], notou-se que isto não influencia no custo da rasterização. Entretando, a presença de texto influencia no tempo de rasterização comparado com uma página em branco. Outro fator analisado que influenciou no tempo de rasterização foi a presença de transparência de texto.

Já para o cálculo do custo de imagens foram analisados dois aspectos: sem reusabilidade e com reusabilidade de imagens. No caso sem reusabilidade de imagens, em um primeiro momento foi considerado a área de ocupação na página. Porém, com os testes realizados em [19], notou-se que isto não influencia no custo da rasterização. Entretanto, a resolução da imagem influencia

no tempo de rasterização, pois os RIPs terão que realizar mais processamento sobre a imagem. Já para o caso com reusabilidade de imagens, a resolução da imagem continua influenciando no tempo de rasterização. Porém, ao adicionar uma imagem reutilizada, o custo desta imagem é menor comparada ao custo de uma imagem não reutilizada. Pois os RIPs buscam na *cache* o resultado da rasterização da imagem reutilizada.

4.2 PDF *Profiler*

Para analisar e fornecer todas as informações contidas em um documento PDF (como, por exemplo, presença de imagens transparentes e imagens reusáveis) foi desenvolvida uma ferramenta chamada PDF *Profiler* [19]. Com o uso desta ferramenta, foi possível realizar a análise do perfil dos *jobs*. A análise realizada concentrou-se no formato PDF por ser este um formato difundido na descrição de documentos personalizados.

Para a execução do PDF *Profiler* deve ser fornecido um arquivo XML de configuração que define as informações que serão analisadas, juntamente com o documento PDF do qual se deseja extrair as informações. Após esta execução, a ferramenta gera um arquivo XML de saída com as informações presente no documento PDF e um arquivo contendo o custo computacional do documento. A seguir tem-se a Figura 4.1 apresentando o funcionamento geral da ferramenta.

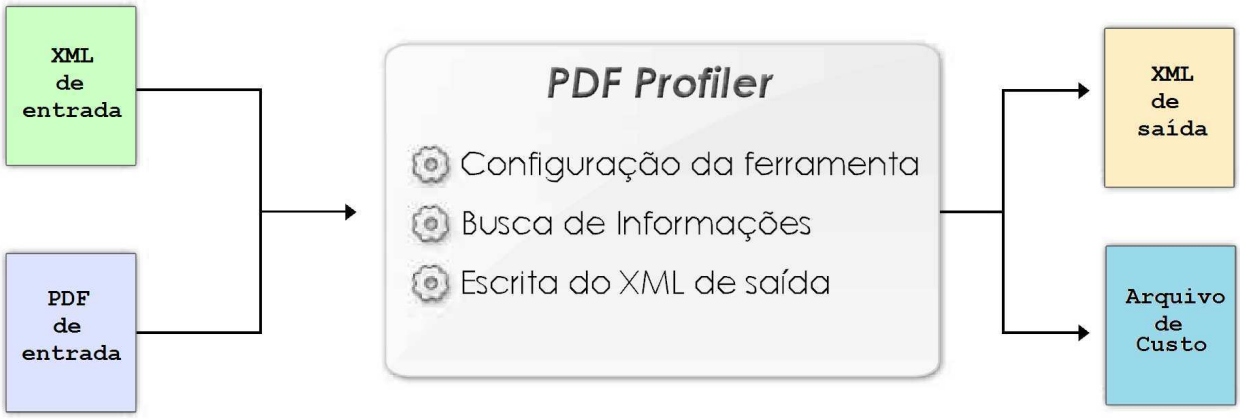

Figura 4.1: Funcionamento geral da ferramenta PDF *Profiler*.

As informações fornecidas de um documento PDF são as páginas (área total), textos (presença em quantas páginas), imagens (área total ocupada com e sem reusabilidade e com transparência), escopo de *image xObjects* (quais páginas estão aplicadas) e transparência de páginas (quais contêm elementos transparentes). A partir das informações fornecidas pelo PDF *Profiler*, pode-se encontrar algumas estratégias para obter um balanceamento de carga mais justo para cada RIP.

4.3 PDF *Splitter*

Em [19] foi desenvolvida uma ferramenta chamada PDF *Splitter*, que tem como objetivo quebrar o documento PDF em vários fragmentos. Assim, é gerado um novo documento PDF para cada fragmento especificado, com suas páginas correspondentes.

Esta ferramenta apresenta um tipo de quebra feita por intervalos. Os fragmentos representam uma quebra do documento original a cada *n* páginas, onde *n* é o intervalo desejado. A seguir tem-se a Figura 4.2 apresentando o funcionamento geral da ferramenta.

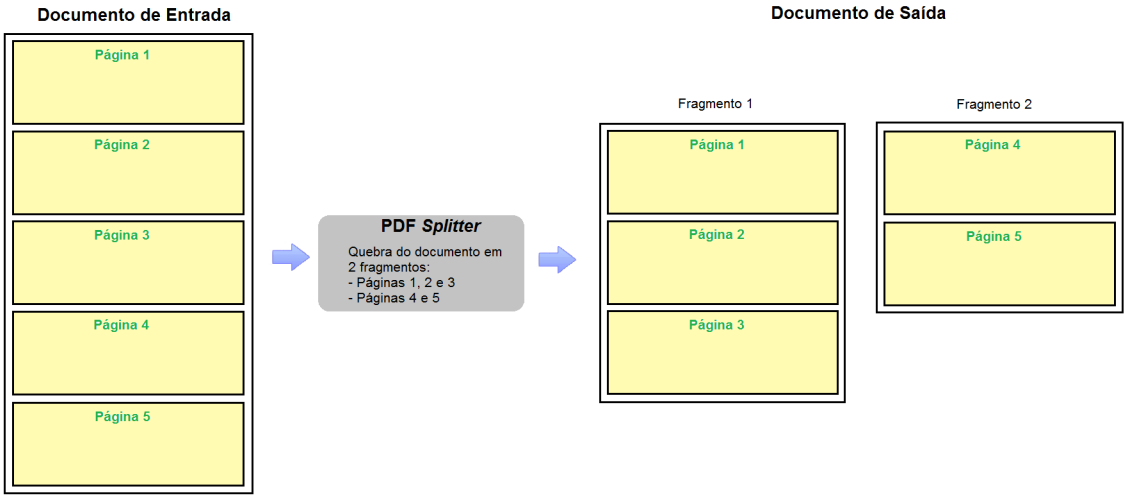

Figura 4.2: Funcionamento geral da ferramenta PDF *Splitter*.

4.4 Escalonadores

Para se obter um desempenho melhor em relação ao algoritmo LS, foram aplicados dois algoritmos de escalonamento: LPT e o *Multifit*. Além destes, um algoritmo com base no LPT foi proposto com algumas mudanças para contornar um problema que poderia ocorrer no escalonamento de *jobs*. Este novo algoritmo foi chamado de LPT Otimizado. Estes três algoritmos foram aplicados na estratégia de rasterização base que apresenta menor número de desvantagem: todos os RIPs por *job*.

Para as estratégias de rasterização base, a eficiência ficou em torno de 82%, enquanto que o algoritmo LPT Otimizado obteve uma eficiência de 92%, o *Multifit* obteve 84% de eficiência e o LPT obteve 88%. Portanto, o LPT Otimizado foi o algoritmo que obteve melhor desempenho entre os outros. A explicação para este resultado é que este algoritmo consegue transmitir imediatamente as tarefas das filas para os RIPs ociosos, paralelamente à análise do perfil dos *jobs* (ferramenta PDF *Profiler*). A seguir está descrito apenas o algoritmo LPT Otimizado (ver descrição do LPT e do *Multifit* na Seção 3.2.1):

O algoritmo LPT Otimizado foi desenvolvido com o intuito de melhorar o algoritmo LPT. Assim, foi estabelecido que a aplicação da métrica sobre as tarefas é realizada de forma concorrente

ao escalonamento das tarefas para os RIPs livres, para poder melhorar o tempo de resposta do escalonador.

Quando o escalonador precisar aplicar a métrica sobre um determinado *job*, será lançada uma nova *thread* responsável por tal função. Anteriormente à criação desta *thread*, as tarefas deste *job* serão inseridas no final da fila, estando disponíveis para serem transmitidas. Assim que as *threads* vão terminando sua função, o custo computacional das tarefas correspondentes será atualizado, removendo-as da fila e inserido-as novamente de forma ordenada.

Com esta estratégia, caso um RIP fique livre e não existam mais tarefas na fila (a não ser aquelas onde o custo computacional não tenha sido calculada), uma destas tarefas será enviada assim mesmo. Caso uma *thread* finalize sua computação e as suas tarefas já tenham sido enviadas, nenhuma ação será tomada.

5. ABORDAGEM PROPOSTA

Este capítulo descreve o trabalho realizado com o objetivo de melhorar o balanceamento de carga entre os RIPs através do desenvolvimento de estratégias e de um Roteador Adaptativo de *Jobs*. Assim, primeiramente é descrita a metodologia deste trabalho, seguidos das estratégias desenvolvidas e do Roteador Adapdativo de *Jobs*.

5.1 Metodologia

As três estratégias base de rasterização de documentos não garantem um bom balanceamento de carga entre os RIPs por apresentar um algoritmo simples (algoritmo LS) para o escalonamento das tarefas (Capítulo 2). No Capítulo 4 foram apresentados novos algoritmos de escalonamento para garantir um melhor balanceamento de carga entre os RIPs. Estes algoritmos conseguiram obter um melhor desempenho em comparação com as três estratégias base de rasterização de documentos. Porém, estes algoritmos ainda podem ser melhorados através da análise mais detalhada das caracaterísticas contidas nos documentos PDF.

A divisão dos *jobs* é feita de maneira simplificada em [19]. Desta forma não se está tendo o aproveitamento máximo que poderia ser obtido. Explorando as caracaterísticas das páginas dos documentos, uma divisão inteligente (que aproveitaria melhor as opções de otimização oferecidas pelos RIPs) pode ser utilizada.

Para atingir o objetivo deste trabalho, em um primeiro momento foi realizada uma modificação na ferramenta PDF *Profiler*, com o intuito de se obter o custo computacional de cada página (anteriormente, o custo computacional era calculado sobre o documento inteiro). Também foi necessário realizar a modificação na ferramenta PDF *Splitter*. Antes, a quebra do documento PDF era feita por intervalos como por exemplo, da página 1 a página 10 e da página 11 a página 20). Com a modificação desta ferramenta, a quebra dos *jobs* é feita de maneira inteligente (páginas específicas, como por exemplo, páginas 1, 7, 8, 9, 11) na tentativa de otimizar o desempenho a partir do tipo de característica contida no documento PDF. Cabe ressaltar que o grão mínimo de cada parte do *job* é uma página. Sendo assim, não poderá ocorrer a existência de um *job* com grão menor que um.

Logo após, cinco estratégias para melhor dividir os *jobs* levando em consideração as características internas dos documentos PDF (transparência e reusabilidade) foram desenvolvidas. Para a realização dos testes das estratégias, foi criado um conjunto de 30 documentos PDFs com características diferentes, reproduzindo a realidade das PSPs. Para a validação destas cinco estratégias testes foram realizados.

Por fim, foi desenvolvido o Roteador Adaptativo de *Jobs*. Para a realização dos testes, foi criado um conjunto de 75 documentos PDF. Para a validação do RAJ testes foram realizados. A seguir serão descritas as estratégias propostas.

5.2 Estratégias

Esta seção apresenta as cinco estratégias propostas neste trabalho com o intuito de obter um bom balanceamento de carga entre os RIPs. No próximo capítulo, são apresentados os modelos de documentos PDF utilizados e os resultados obtidos utilizando estas estratégias.

5.2.1 Transparência

A característica que mais influencia no tempo de processamento de um *job* é a presença ou não de imagens transparentes. Enquanto um documento de 100 páginas contendo uma imagem "X" sem transparência em cada página leva 21100 segundos para rasterizar, um documento de 100 páginas contendo a mesma imagem "X" porém, transparente em cada página, leva 26400 segundos, por exemplo. A partir disso, desenvolveu-se um algoritmo que leva em consideração essa característica.

O algoritmo separa primeiramente as páginas transparentes, ordenando-as pelo seu custo computacional (maior custo para o menor) em um fila de páginas. Este custo computacional é obtido com o uso da ferramenta PDF *Profiler*, que foi modificada para obter o custo computacional de cada página. Após a ordenação destas páginas transparentes, as páginas restantes do *job* (páginas com imagens opacas e/ou texto) também são ordenadas pelo seu custo computacional para, então, serem inseridas na fila de páginas. Quando todas as páginas do *job* forem inseridas nesta fila de páginas, começa a ser feita a distribuição de acordo com a quantidade de RIPs (por exemplo, se houver 4 RIPs serão criados 4 fragmentos) de forma que os RIPs fiquem com uma carga similar de transparências. Esta distribuição é conhecida como "zig-zag". A Figura 5.1 apresenta um exemplo dessa distribuição das páginas de maneira balanceada.

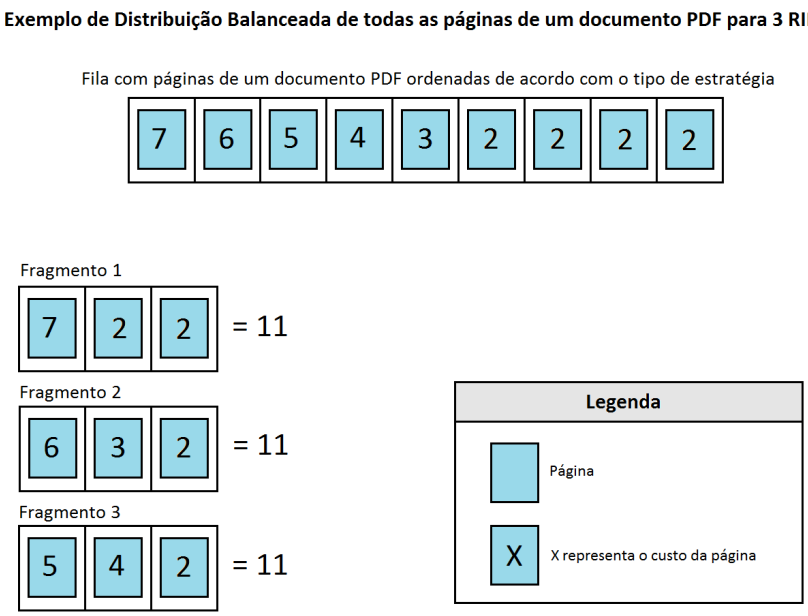

Figura 5.1: Distribuição balanceada das páginas de um documento PDF de acordo com a estratégia.

Como nota-se, a fila com as páginas do documento PDF está ordenada de acordo com a estratégia (nesse caso, estratégia Transparência, onde o documento PDF contém 5 páginas, com imagens transparentes e 4 páginas com texto). Quando esta fila de páginas estiver pronta (todas as páginas do documento inseridas), começa a ser realizada a distribuição de maneira balanceada. Esta distribuição é feita da seguinte forma: associam-se as páginas uma a uma para cada fragmento (onde o número de fragmentos é igual ao número de RIPs) de forma "zig-zag", ou seja, a primeira página para o fragmento 1, a segunda para o fragmento 2, a terceira para o fragmento 3, a quarta também para o fragmento 3, a quinta para o fragmento 2, a sexta para o fragmento 1 e assim sucessivamente. Porém, a partir do ponto onde todos os fragmentos receberam uma página e ao tentar receber uma página pela segunda vez, um teste começa a ser realizado. Este teste tem o objetivo de não deixar inserir mais páginas caso o custo atual do fragmento somado ao custo da página que será inserida for maior do que o custo que o fragmento deveria ter (custo total do documento PDF dividido pelo número de fragmentos). Desta maneira, evita-se que o fragmento fique desbalanceado.

Quando se tenta inserir uma tarefa em um fragmento e o custo computacional atual do fragmento somado ao custo da página que será inserida for maior do que o custo computacional que deveria ter, esta tarefa passa para o próximo fragmento (obedecendo a ordem da distribuição) e realiza-se o mesmo teste. Isto se repete até que se tenha tentado inserir em todos os fragmentos. Caso não seja possível inserir em algum fragmento, a inserção é feita através da "força-bruta", inserindo a página no fragmento que apresentar o menor custo computacional.

Nota-se que após a distribuição ser realizada, o custo computacional total do fragmento ficou balanceada (cada fragmento com custo computacional igual a 11). A seguir tem-se um exemplo para um melhor entendimento da estratégia Transparência, que pode ser acompanhado com a Figura 5.2.

Seja um *job* com 10 páginas que deve ser distribuído entre 2 RIPS, onde as 4 primeiras páginas do *job* contêm imagens transparentes com custo computacional de, respectivamente, 7, 10, 8, 9 e as próximas 6 páginas deste *job* contêm apenas texto possuindo um custo computacional igual a 2 (páginas contendo apenas texto sempre possuem o mesmo custo). O algoritmo começa ordenando as páginas com imagens transparentes e inserindo-as na fila de páginas: página 2 (custo 10), página 4 (custo 9), página 3 (custo 8) e página 1 (custo 7). Em seguida, as páginas restantes do *job* são ordenadas (nesse caso como as próximas páginas contêm apenas texto elas não precisam ser ordenadas, pois seu custo computacional são sempre iguais, independente do número de linhas que tiver) e também inseridas na fila de páginas. Após este processo, começa ser feita a distribuição destas páginas entre os fragmentos de acordo com a quantidade de RIPs. A distribuição final fica, então, 2 páginas contendo imagens transparentes e 3 páginas contendo texto para os fragmentos 1 e 2. Deste modo, o processamento é otimizado, tendo um ganho significativo de tempo na execução do *job*.

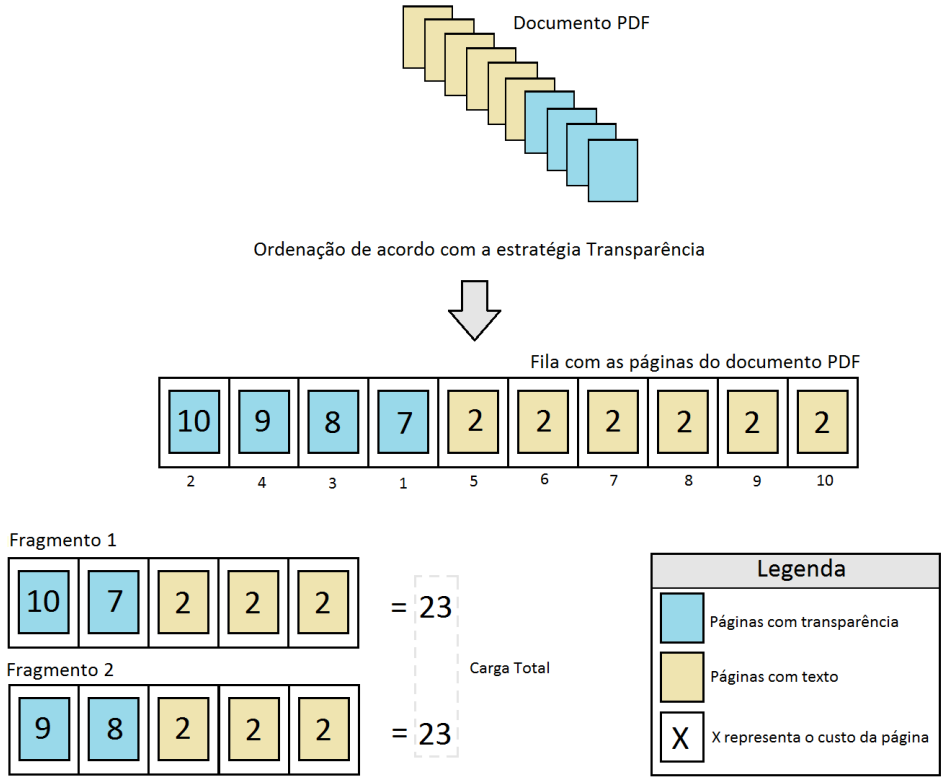

Figura 5.2: Distribuição balanceada das páginas de um *job* na estratégia Transparência.

5.2.2 Reusabilidade

Quando imagens com reusabilidade estão agrupadas, seu tempo de processamento diminui (a primeira página continua com o mesmo custo computacional, porém as páginas restantes tem seu custo computacional reduzido), pois os RIPs fazem *cache* destas imagens e, quando encontradas pela segunda vez no documento, não precisam ser rasterizadas novamente. Se as imagens estiverem em documentos ou em RIPs diferentes, serão processadas diversas vezes. Com isto, desenvolveu-se um algoritmo levando em consideração esta característica.

O algoritmo proposto primeiramente une as imagens com reusabilidade, formando então um conjunto de imagens reusáveis. Isto é feito até que todos os conjuntos com imagens reusáveis estejam criados. Logo após, é feita uma ordenação pelo custo computacional do conjunto (custo maior até o menor) e estes conjuntos são inseridos na fila de páginas.

Caso o número de conjunto com imagens reusáveis seja menor do que o número de RIPs, cada conjunto é quebrado em dois (caso ainda seja menor o conjunto, é quebrado em dois novamente até que o número de conjunto seja maior do que o número de RIPs). Após acabar esta primeira etapa, ordena-se também as páginas restantes (páginas com imagens opacas e/ou texto) do *job* para também inserir na fila de páginas. Por fim, é realizada a distribuição das páginas entre os fragmentos.

A distribuição das páginas nesta estratégia segue a distribuição utilizada na estratégia Transparência (distribuição "zig-zag" mostrada na Figura 5.3). A seguir, é apresentado um exemplo para

um compreendimento da estratégia Reusabilidade, que pode ser acompanhado na Figura 5.3.

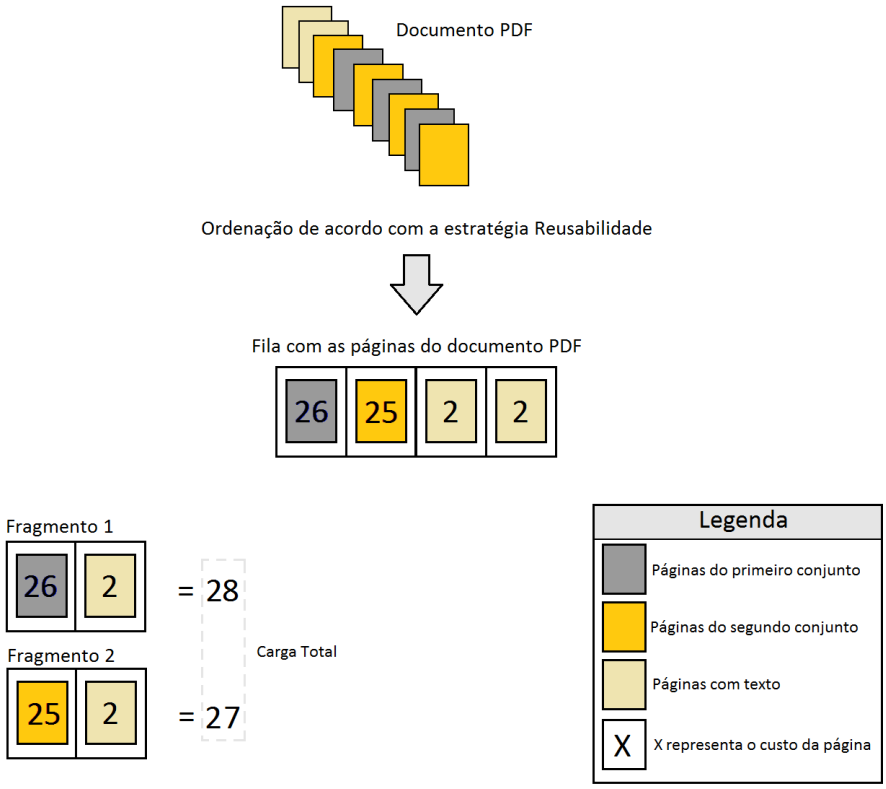

Figura 5.3: Distribuição balanceada das páginas de um *job* na estratégia Reusabilidade.

Seja um *job* contendo 9 páginas que deve ser distribuído entre 2 RIPS, onde nas 3 primeiras páginas pares contêm a mesma imagem (imagem A), e o custo computacional da cada página seja igual a 10, nas primeiras 4 páginas ímpares também contêm imagens iguais (imagens B) com custo computacional de cada página igual a 7 e as 2 páginas restantes contêm apenas texto com custo computacional igual a 2. A distribuição seria: as 3 primeiras páginas pares formariam um conjunto. Este primeiro conjunto teria um custo computacional total de 26, pois como o custo da página é igual a 10 e quando se agrupam imagens com reusabilidade o custo computacional das páginas seguintes tem seu custo computacional reduzido, neste caso as 2 páginas ficariam com um custo computacional igual a 8 cada uma (hipoteticamente). As próximas 4 páginas ímpares formariam um segundo conjunto. Este segundo conjunto teria um custo computacional igual a 25, pois como o custo da página é igual a 7 e como descrito anteriormente, que quando se agrupam imagens com reusabilidade o custo computacional das páginas seguintes tem seu custo computacional reduzido, neste caso as 3 páginas ficariam com um custo computacional igual a 6 cada uma. Com os conjuntos criados, insere-se na fila de páginas os conjuntos ordenados pelo seu custo computacional. Após isto, como as próximas 2 páginas restantes contêm apenas texto, elas não necessitam ser ordenadas pois seus custos computacional são iguais. Assim, elas também são inseridas na fila de páginas. Por fim, realiza-se a distribuição entre os fragmentos.

5.2.3 Mais Transparência e Menos Reusabilidade

Há uma grande variedade de tipos de documentos PDF (documentos com mais páginas, menos páginas, apresentando apenas uma característica, apresentando várias caractarísticas em conjunto). Para os documentos PDF que apresentam a característica de transparência em conjunto com a de reusabilidade, duas estratégias foram propostas.

A primeira estratégia seria quando se tem mais imagens com transparência do que imagens com reusabilidade. A segunda estratégia seria quando o documento apresenta mais imagens com reusabilidade do que imagens com transparência. Esta seção aborda a primeira estratégia (Mais Transparência e Menos Reusabilidade) e na seção seguinte é descrita a segunda estratégia (Menos Transparência e Mais Reusabilidade). Para a estratégia de Mais Transparência e Menos Reusabilidade um algoritmo foi proposto. A seguir tem-se o algoritmo:

Como as páginas mais relevantes em um documento PDF são as que contêm imagens transparentes, estas, são ordenadas primeiramente pelo seu custo computacional e logo após são inseridas na fila de páginas. Logo após, unem-se todos as páginas contendo reusabilidade formando conjuntos. Estes conjuntos são ordenados pelo seu custo computacional para então serem inseridos na fila de páginas. Assim que a fila com as páginas do documento estiver pronta, inicia-se a distribuição para os fragmentos. Após inserir as páginas com imagens transparentes e ao tentar adicionar um conjunto em um fragmento e seu custo computacional somando ao custo computacional atual do fragmento for maior do que o custo que o fragmento deveria ter (custo total do documento PDF dividido pelo número de RIPs), este fragmento então é particionado em dois e é feito o teste novamente, até que o conjunto seja inserido ou no primeiro fragmento da tentativa de inserção ou no fragmento de menor custo computacional).

A distribuição balanceada utilizada nesta estratégia também segue a distribuição balanceada descrita na Seção 5.2.1 (estratégia Transparência). A seguir, tem-se um exemplo para um entendimento melhor desta estratégia e que pode ser acompanhado na Figura 5.4.

Seja um *job* contendo 10 páginas que deve ser distribuído entre 2 RIPS, onde nas páginas 1, 3, 5, 7 e 9 (com custo computacional igual a 8, 5, 3, 6 e 4) contêm imagens transparentes e nas páginas 2 e 4 contém a mesma imagem (imagem A) com custo computacional 5 e nas páginas 6, 8 e 10 possuem páginas com texto (custo computacional igual a 2). Primeiramente, as páginas com transparências são ordenadas pelo seu custo computacional e são inseridas na fila de páginas. Logo em seguida, as páginas contendo imagens reusáveis são agrupadas em um conjunto e ordenadas pelo seu custo computacional. Este conjunto é inserido na fila de páginas. Após, as páginas com texto também são inseridas na fila de páginas. Quando todas as páginas forem inseridas na fila, a distribuição começa ser feita. Sempre antes de inserir um conjunto com reusabilidade nos fragmentos, realiza-se o teste para saber se o custo atual do fragmento somado ao custo computacional do conjunto é maior do que o custo que o fragmento deveria ter. Se caso for maior divide-se o conjunto em dois, senão, insere-se o conjunto.

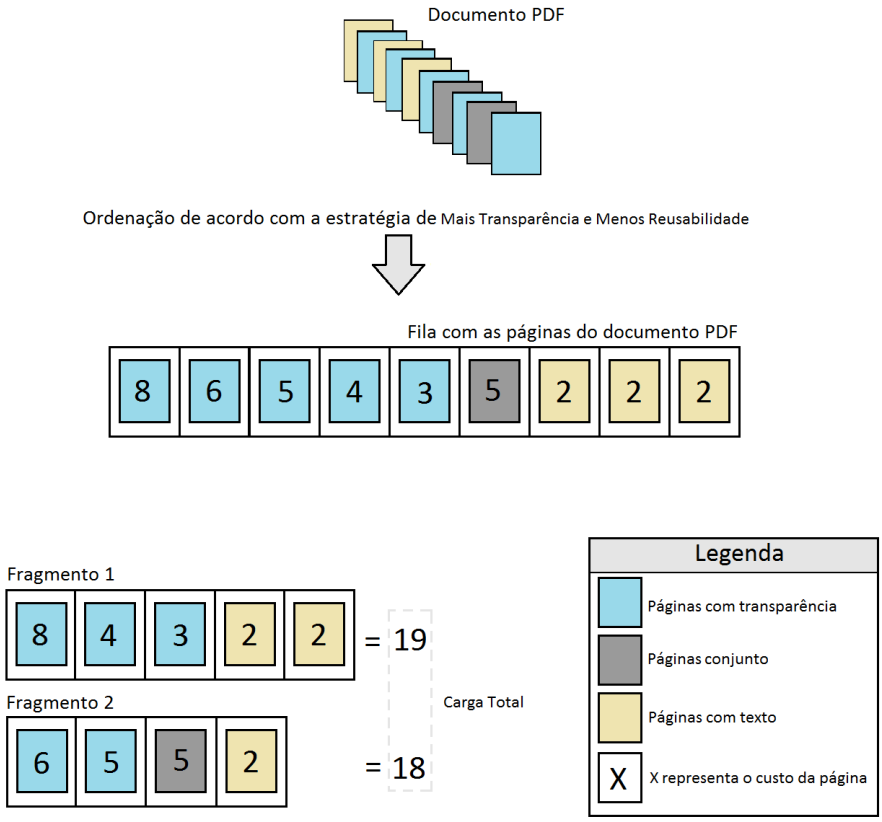

Figura 5.4: Distribuição balanceada das páginas de um *job* na estratégia Mais Transparência e Menos Reusabilidade.

5.2.4 Menos Transparência e Mais Reusabilidade

Para a estratégia de Menos Transparência e Mais Reusabilidade foi proposto um algoritmo que está logo a seguir. Como o documento PDF contém mais páginas com imagens reusáveis do que com transparência, une-se todas as páginas com imagens reusáveis, formando conjuntos. Após a formação, é feita a ordenação através dos custos computacionais dos conjuntos, para então serem inseridos na fila de páginas. Logo em seguida, as páginas com imagens transparentes são ordenadas também pelo seu custo computacional para então serem inseridas na fila de páginas. Por fim, ordena-se as páginas restantes e logo após insere-as na fila de páginas. Quando todas as páginas tiverem sido inseridas na fila, começa a ser realizada a distribuição entre os fragmentos.

A distribuição de maneira balanceada utilizada nesta estratégia segue o mesmo modelo adotado nas outras estratégias, ou seja, a distribuição "zig-zag" mostrada da Figura 5.1. A seguir, um exemplo é descrito para compreender melhor esta estratégia, que pode ser acompanhada na Figura 5.5.

Seja um *job* contendo 8 páginas que deve ser distribuído entre 2 RIPS, onde nas páginas 1, 3 e 5 contêm a mesma imagem (imagem A) com custo computacional 7, nas páginas 2, 4 e 6 contêm a mesma imagem (imagem B) com custo computacional 8 e nas páginas 7 (custo computacional 10) e 8 (custo computacional 6) contêm imagens transparentes. Primeiramente, as páginas com

imagens reusáveis são agrupadas em conjuntos (2 conjuntos, o primeiro com custo computacional igual a 22 e o segundo igual 19). Estes conjuntos são ordenados pelo seu custo computacional para logo em seguida serem inseridos na fila de páginas. Logo após, ordena-se as páginas transparentes e insere-as na fila de páginas. Por fim, a distribuição entre os fragmentos começa a ser realizada.

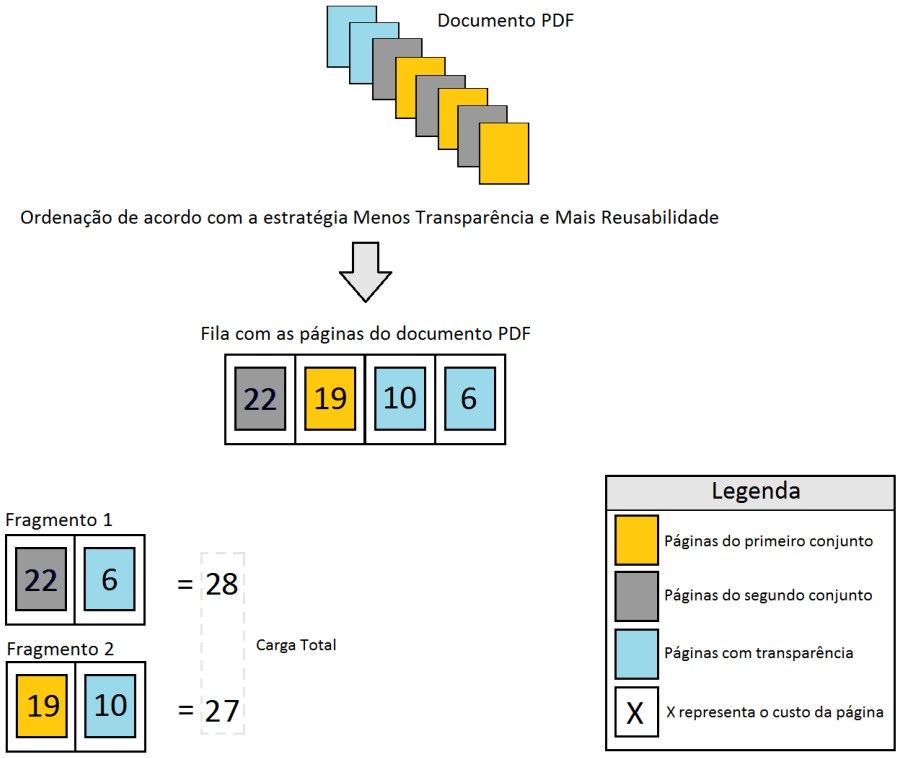

Figura 5.5: Distribuição balanceada das páginas de um *job* na estratégia Menos Transparência e Mais Reusabilidade.

5.2.5 Sem Transparência e Sem Reusabilidade

Os documentos PDFs poderão apresentar páginas sem imagens com transparência e páginas sem imagens com reusabilidade. Através deste cenário, foi proposto um algoritmo para satisfazer esta condição.

O algoritmo começa ordenando todas as páginas pelo seu custo computacional (do maior ao menor). Em seguida, é realizada a inserção na fila de páginas e a distribuição de maneira balanceada entre os fragmentos é iniciada. A ditribuição balanceada utilizada nesta estratégia também segue a mesma distribuição utilizada nas outras estratégias (distribuição "zig-zag") e é apresentada na Figura 5.1. A seguir tem-se um exemplo para a melhor compreensão desta estratégia, que pode ser acompanhada na Figura 5.6.

Seja um *job* contendo 8 páginas que deve ser distribuído entre 2 RIPS, onde nas páginas 1, 3, 4 e 7 (custo computacional 6, 5, 7 e 8 respectivamente) apresentam imagens diferentes e opacas e nas páginas 2, 5, 6 e 8 contêm apenas texto. Todas as páginas são ordenadas pelo seu custo computacional. Após é feita a distribuição de todas as páginas entre os fragmentos.

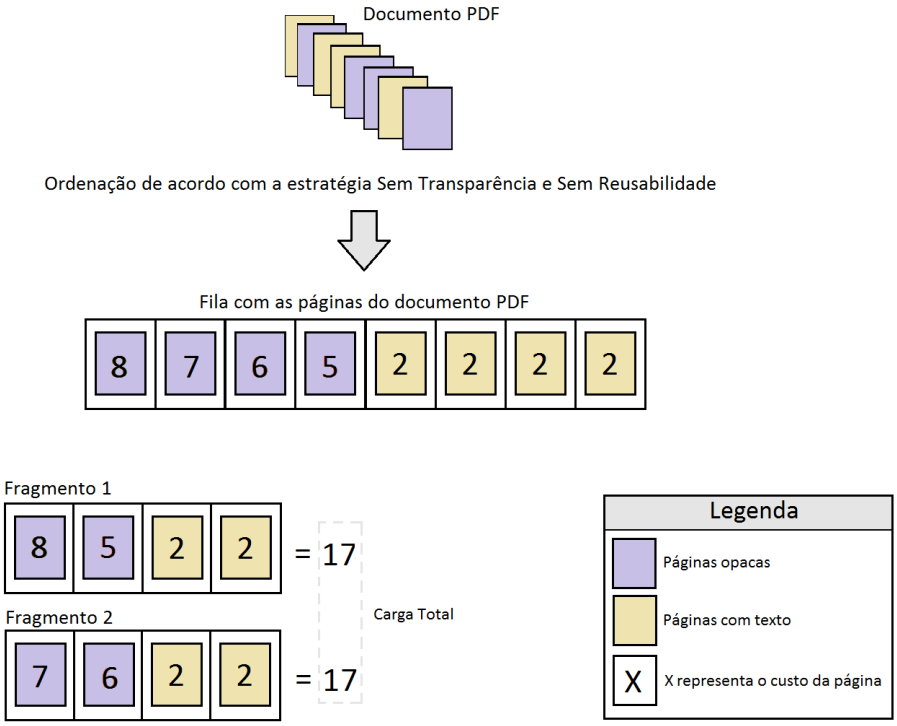

Figura 5.6: Distribuição balanceada das páginas de um *job* na estratégia Sem Transparência e Sem Reusabilidade.

5.3 Roteador Adaptativo de *Jobs* - RAJ

O Roteador Adaptativo de *Jobs*, tem a capacidade de decidir a partir das informações contidas em um arquivo XML, qual das estratégias desenvolvidas será utilizada por um determinado *job* para gerar os fragmentos que serão rasterizados. A Figura 5.7, apresenta a visão geral do processo de rasterização.

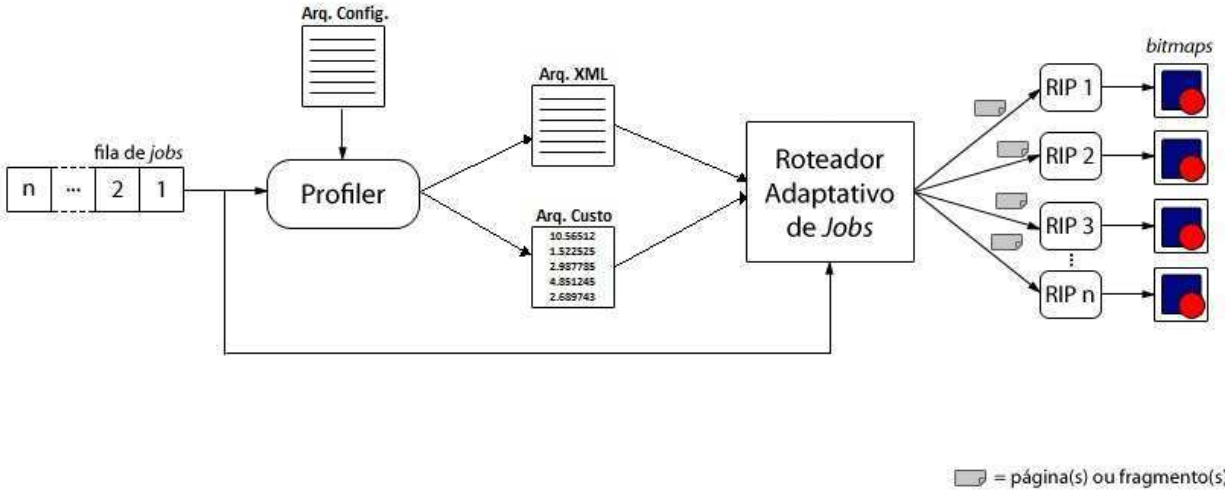

Figura 5.7: Visão geral do processo de rasterização com a inclusão do módulo RAJ.

Em um primeiro momento, tem-se a fila com todos os *jobs* que passarão pelo processo de rasterização. Após, cada um dos *jobs* é enviado para a ferramenta PDF *Profiler*, que tem como arquivo de entrada um arquivo de configuração (Figura 5.8) contendo quais informações do *job* serão obtidas. Esta ferramenta obtem todas as informações, ou as informações mais relevantes contidas nestes *jobs*.

```
<?xml version="1.0"?>

<!DOCTYPE profiler SYSTEM "profiler.dtd">

<profiler>
     <fragmento>
          <areaPagina>SIM</areaPagina>
          <areaImagemReusavel>SIM</areaImagemReusavel>
          <areaImagemTranspReusab>SIM</areaImagemTranspReusab>
          <areaImagemNaoReusab>SIM</areaImagemNaoReusab>
          <areaImagemTranspNaoReusab>SIM</areaImagemTranspNaoReusab>
          <presencaTexto>SIM</presencaTexto>
          <presencaTextoTransp>SIM</presencaTextoTransp>
     </fragmento>
</profiler>
```

Figura 5.8: Arquivo de configuração do PDF *Profiler*.

O PDF *Profiler*, ao analisar o perfil dos *jobs*, gera um arquivo XML (Figura 5.9) contendo as informações de cada página (quais contêm texto, quais contêm imagens, se estas imagens são opacas, transparentes ou reusáveis) e um arquivo contendo o custo computacional total do documento e o custo computacional de cada página (Figura 5.10). Quando estes arquivos são gerados, o Roteador Adaptativo de *Jobs* analisa estes arquivos e, através deles, consegue descobrir qual das estratégias deve ser utilizada em um determinado *job*.

```
<?xml version="1.0" encoding="UTF-8"?>
<profile>
   <infoPaginas>
      <pagina cor="SIM" numero="1" transparente="SIM"/>
      <pagina cor="SIM" numero="2" transparente="SIM"/>
      <pagina cor="SIM" numero="3" transparente="SIM"/>
   </infoPaginas>
   <fragmento numero="1">
      <areaPagina>500990.00</areaPagina>
      <imagemReusavel opaca="0.00" transparente="0.00"/>
      <imagemNaoReusavel opaca="0.00" transparente="149264.00"/>
      <texto paginasOpaca="0" paginasTransp="0"/>
   </fragmento>
   <infoObjeto>
      <objeto id="79" pagina="25"/>
      <objeto id="51" pagina="16"/>
      <objeto id="8"  pagina="2"/>
      <objeto id="23" pagina="7"/>
      <objeto id="48" pagina="15"/>
 </infoObjeto>
</profile>
```

Figura 5.9: Modelo de um arquivo XML gerado pelo PDF *Profiler*.

A Figura 5.11 apresenta o funcionamento detalhado do módulo RAJ. Ao receber o arquivo XML e o arquivo Custo, o bloco Analisador de Perfil dos *Jobs*, lê o arquivo XML para descobrir qual estratégia será utilizada em cima de um *Job*. Este analisador também lê o arquivo Custo para criar uma lista de informações com as páginas e seus respectivos custos.

```
3.908870765487394955
0.009977510707787436
0.009148112915416331
0.009289212538163292
0.008763733985697174
0.009057093134582382
0.009178035668365492
        .
        .
        .
0.009178035668365492
0.008727625480537392
```

Figura 5.10: Modelo de um arquivo Custo gerado pelo PDF *Profiler*.

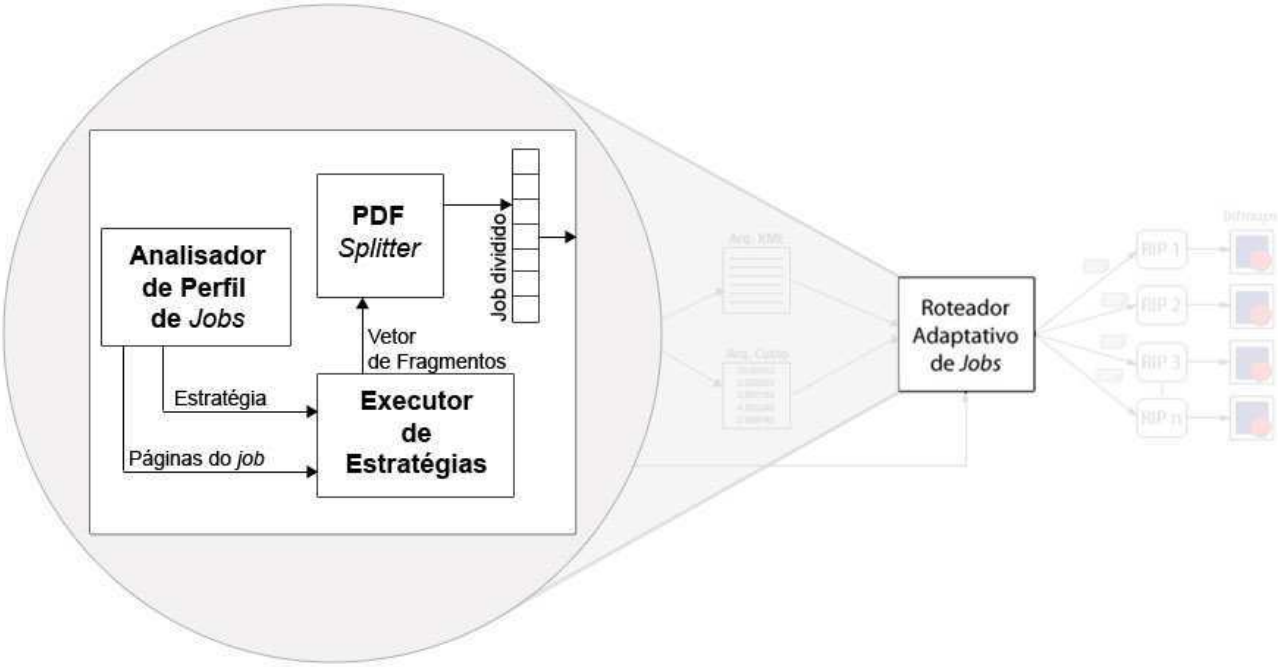

Figura 5.11: Funcionamento do módulo RAJ.

Ao descobrir qual é a estratégia, o bloco Analisador envia a estratégia definida para o próximo bloco assim como a lista contendo as informações do *job* (lista de páginas). O bloco Executor de Estratégias, ao receber a saída do Analisador, começa analisando a lista de informações para então criar o vetor com os fragmentos (número de fragmento igual ao número de RIPs) de acordo com a estratégia recebida. Assim que os fragmentos forem criados (com as páginas no novo *job*) estes são enviados para a ferramenta PDF *Splitter*. Após a ferramenta particionar o *job* de maneira inteligente (ver exemplo na Seção 5.2.1), os novos *jobs* são inseridos em um fila de tarefas para então serem enviados para os RIPs disponíveis.

6. AVALIAÇÃO DAS ESTRATÉGIAS

Este capítulo apresenta o ambiente de teste utilizado para executar as cinco estratégias propostas. Também são apresentados os casos de testes criados com diferentes características representando casos da realidade das PSPs. Por fim, tem-se os resultados obtidos com o uso das estratégias desenvolvidas comparando-as com o algoritmo que obteve o melhor desempenho em [19]. Este algoritmo foi o LPT Otimizado. Os resultados foram obtidos através de uma média de 20 execuções com o uso de um RIP *open-source*, o *ImageMagick converter* [12]. Nos testes, os documentos PDFs foram rasterizados utilizado-se 40 DPI (*Dots per Inch*) de resolução.

6.1 Ambiente de Teste

Será descrito nesta seção o ambiente de *hardware* e de *software* utilizado para criar os *jobs* e executar as cinco estratégias desenvolvidas. Além disso, serão apresentados os *jobs* utilizados para realização dos testes.

6.1.1 Ambiente de *Hardware* e *Software*

Para se aproximar da realidade das PSPs, o ambiente de *hardware* que foi utilizado para realização dos testes é um agregado de 6 *blades* HP ProLiant BL20p (arquitetura multicomputador, podendo ser classificada como um COW - *Cluster Of Workstations*), na qual cada máquina contém dois processadores *Single-Core Intel Xeon* 3.6GHz (FSB 800MHz, 2MB L2), com 2GB de memória DDR2 SRAM (400MHz), rodando sob um sistema operacional Linux. Além disso estas *blades* estão conectadas através de uma rede de alta velocidade *Gigabit Ethernet*.

Para a criação do conjunto de *jobs*, utilizou-se a biblioteca Java iText na versão 2.0.7 [13]. Esta biblioteca é *Open Source* e está muito bem documentada facilitando seu uso. A partir dela é possivel gerar documentos contendo textos, tabelas, imagens e apresenta diversos tipos de fontes.

6.1.2 Casos de Testes

Para avaliar as cinco estratégias desenvolvidas, um conjunto de 30 *jobs* com características diferentes foi criado. Os *jobs* criados estão divididos em cinco grupos: PDFs com transparências, PDFs com reusabilidade, PDFs com mais transparência do que reusabilidade, PDFs com mais reusabilidade do que transparência e PDFs sem a presença de transparência e de reusabilidade. A Tabela 6.1 mostra as características presentes nos *jobs* como o número de documentos, número de total de páginas, o número total de imagens e o número total de páginas com textos.

Tabela 6.1: *Características dos jobs utilizados nos testes das estratégias*

Job	Documentos	Páginas	Imagens	Páginas com Textos
Transp 1	1	9	4	9
Transp 2	1	15	6	15
Transp 3	1	9	3	9
Reusab 1	150	300	300	300
Reusab 2	50	200	100	200
Reusab 3	800	200	800	200
Reusab 4	1200	300	1200	300
Reusab 5	150	450	450	450
Reusab 6	150	300	150	300
Reusab 7	100	200	200	200
Reusab 8	250	500	250	500
Reusab 9	100	600	600	600
Reusab 10	70	700	490	700
Reusab 11	200	200	400	200
Reusab 12	300	300	300	300
MaisTMenosR 1	100	100	38	100
MaisTMenosR 2	80	80	31	80
MaisTMenosR 3	90	90	43	90
MaisTMenosR 4	120	120	45	120
MaisTMenosR 5	100	100	40	100
MenosTMaisR 1	100	100	34	100
MenosTMaisR 2	80	80	36	80
MenosTMaisR 3	90	90	30	90
MenosTMaisR 4	120	120	30	120
MenosTMaisR 5	100	100	34	100
SemTSemR 1	100	100	20	100
SemTSemR 2	120	120	25	120
SemTSemR 3	100	100	28	100
SemTSemR 4	80	80	17	80
SemTSemR 5	60	60	17	60

Pode-se notar que os documentos criados possuem uma grande diversidade de número de páginas, textos e imagens retratando a realidade das PSPs. Desta maneira, será possível avaliar o comportamento das estratégias.

6.2 Análise dos Resultados das Estratégias

Para a realização dos testes foi feita uma comparação com o algoritmo que obteve o melhor desempenho das estratégias desenvolvidas em [19]. Este algoritmo foi o LPT Otimizado. A seguir são apresentados os resultados para as cinco estratégias propostas.

6.2.1 Transparência

Estabeleceu-se um conjunto de 3 *jobs* com diferentes características para validar a estratégia de transparência descrita no Capítulo 5. O conjunto de *jobs* selecionados foram: Transp 1, Transp 2 e Transp 3 citados anteriormente. Nesta estratégia foi considerado 3 RIPs por *job*. A Figura 6.1 mostra o gráfico referente a distribuição em 3 RIPs com tempo de processamento em segundos.

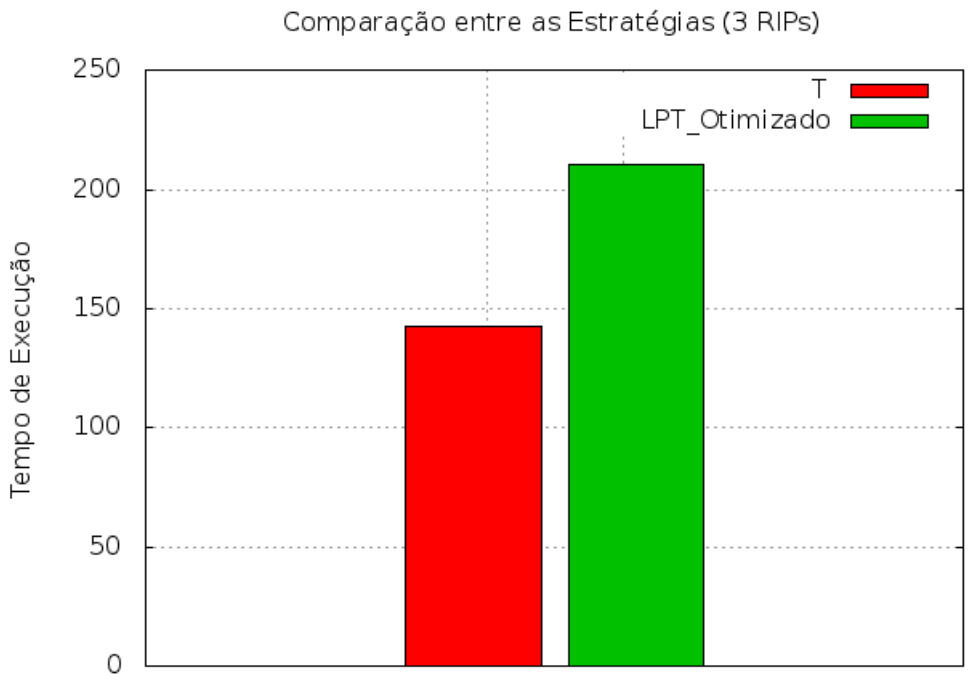

Figura 6.1: Gráfico de comparação entre a estratégia Transparência (T) e LPT Otimizado com uma distribuição de 3 RIPs.

Como pode-se notar, a estratégia de Transparência mostrou-se mais eficiente do que o LPT Otimizado. Enquanto o LPT Otimizado rasterizou os documentos PDF em 143 segundos, a estratégia de transparência levou apenas 84 segundos para realizar a rasterização. Com isto, o ganho foi de 32,21%.

6.2.2 Reusabilidade

Para validar esta estratégia implementada com a característica de reusabilidade de imagens (Capítulo 5), estabeleceu-se um conjunto de 12 *jobs* (Reusab 1, Reusab 2, Reusab 3, Reusab 4, Reusab 5, Reusab 6, Reusab 7, Reusab 8, Reusab 9, Reusab 10, Reusab 11 e Reusab 12 descritos anteriormente) com diferentes características. Os *jobs* criados para a realização dos testes possuem grande variedade de quantidade de textos e imagens, retratando assim, a realidade da maioria das PSPs.

Para a execução dos testes o número de RIPs utilizados foi de 3. O gráfico da Figura 6.2 é referente ao processamento (em segundos) dos documentos PDFs citados acima, para uma distribuição de 3 RIPs por *job*.

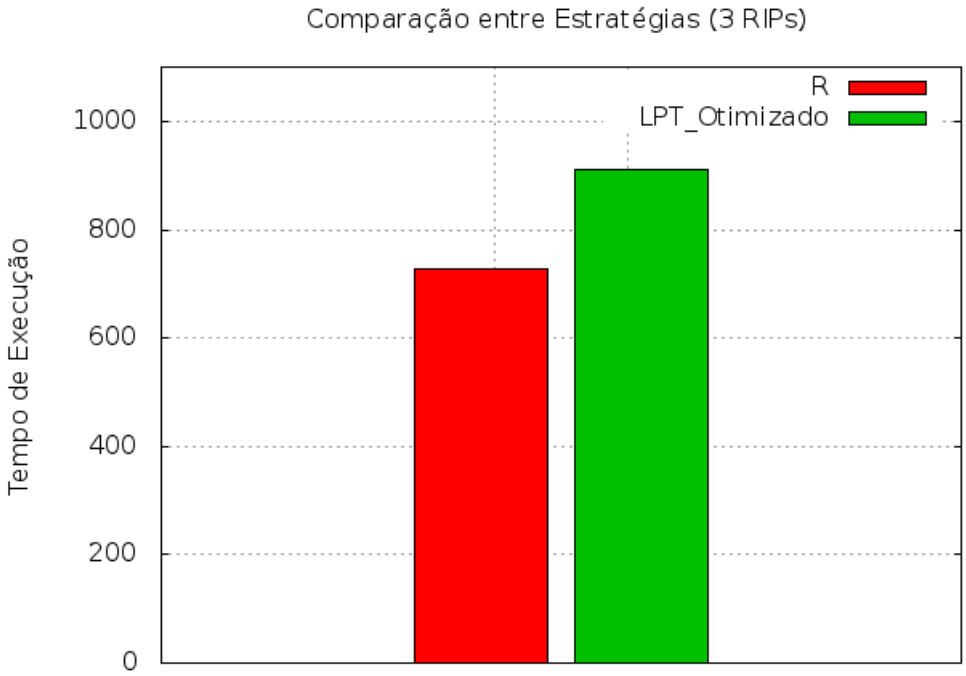

Figura 6.2: Gráfico de comparação entre a estratégia Reusabilidade (R) e LPT Otimizado com uma distribuição de 3 RIPs.

Nota-se que com a distribuição de 3 RIPs a estratégia de reusabilidade se mostrou um pouco mais eficiente novamente do que o algoritmo LPT Otimizado. Enquanto o LPT Otimizado rasterizou os documentos PDF em 911 segundos, a estratégia de reusabilidade levou apenas 727 segundos para realizar a rasterização. Assim, o ganho foi de 20,2%.

6.2.3 Mais Transparência e Menos Reusabilidade

Um conjunto de 5 *jobs* (MaisTMenosR 1, MaisTMenosR 2, MaisTMenosR 3, MaisTMenosR 4 e MaisTMenosR 5) com diferentes características foi utilizado para validar a estratégia Mais transparência e Menos reusabilidade descrita no Capítulo 5. Nesta estratégia foi considerado 3 RIPs por *job*. A figura 6.3 mostra o gráfico referente a distribuição em 3 RIPs.

Nota-se que esta estratégia também foi mais eficiente do que o LPT Otimizado utilizando uma distribuição para 3 RIPs. Enquanto o LPT Otimizado rasterizou os documentos PDF em 246,8 segundos, a estratégia Mais Transparência e Menos Reusabilidade levou apenas 151,6 segundos para realizar a rasterização. Assim, o ganho foi de 38,57%.

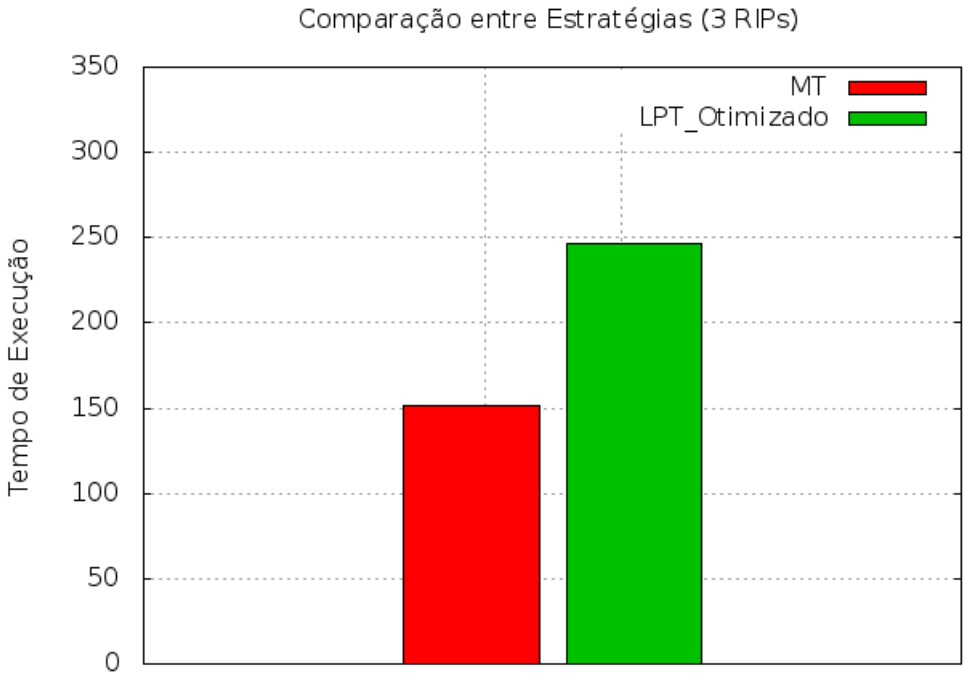

Figura 6.3: Gráfico de comparação entre a estratégia Mais Transparência e Menos Reusabilidade (MT) e LPT Otimizado com uma distribuição de 3 RIPs.

6.2.4 Menos Transparência e Mais Reusabilidade

Um conjunto de 5 *jobs* com diferentes características foi utilizado para validar a estratégia Menos Transparência e Mais Reusabilidade descrita no Capítulo 5. Os *jobs* utilizados foram: (MenosTMaisR 1, MenosTMaisR 2, MenosTMaisR 3, MenosTMaisR 4 e MenosTMaisR 5. Nesta estratégia foi considerado 3 RIPs por *job*. A Figura 6.4 mostra o gráfico referente a distribuição em 3 RIPs.

Com uma distribuição de 3 RIPS, nota-se que esta estratégia obteve um desempenho melhor do que o LPT Otimizado. Enquanto o LPT Otimizado rasterizou os documentos PDF em 267,8 segundos, a estratégia Menos Transparência e Mais Reusabilidade levou apenas 157 segundos para realizar a rasterização. Assim, o ganho foi de 41,37%.

6.2.5 Sem Transparência e Sem Reusabilidade

Um conjunto de 5 *jobs* (SemTSemR 1, SemTSemR 2, SemTSemR 3, SemTSemR 4 e SemTSemR 5) com diferentes características foi utilizado para validar a estratégia Sem Transparência e Sem Reusabilidade descrita no Capítulo 5. Nesta estratégia foi considerado 3 RIPs por *job*. A Figura 6.5 mostra o gráfico referente a distribuição em 3 RIPs.

Nesta estratégia também nota-se que com uma distribuição de 3 RIPs o desempenho em relação ao LPT Otimizado foi melhor. Enquanto o LPT Otimizado rasterizou os documentos PDF em 119,7 segundos, a estratégia Sem Transparência e Sem Reusabilidade levou apenas 106 segundos para realizar a rasterização. Assim, o ganho foi de 11,45%.

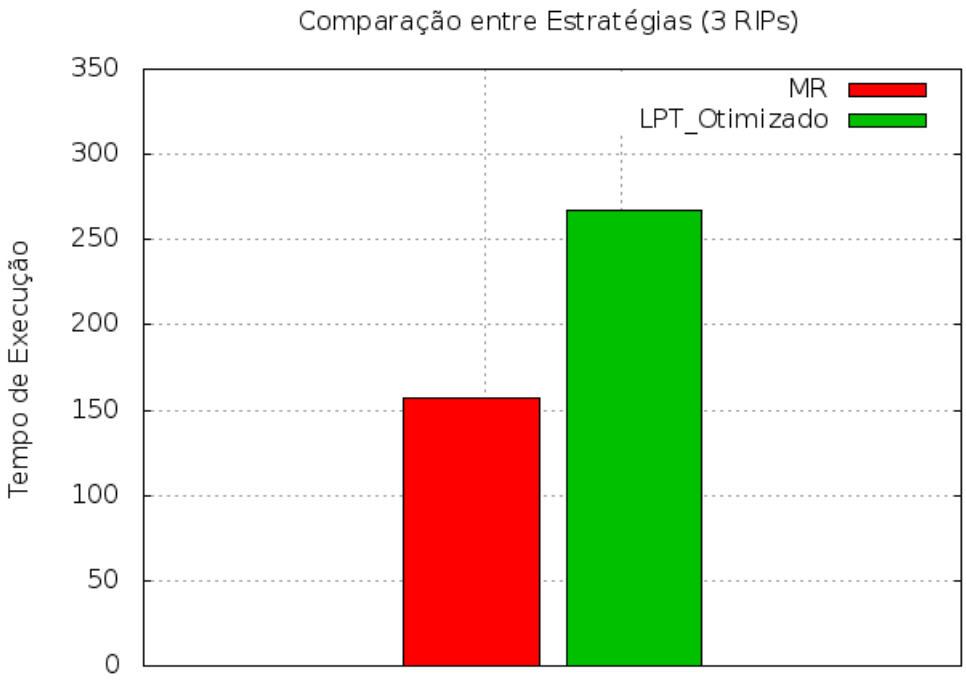

Figura 6.4: Gráfico de comparação entre a estratégia Menos Transparência e Mais Reusabilidade (MR) e LPT Otimizado com uma distribuição de 3 RIPs.

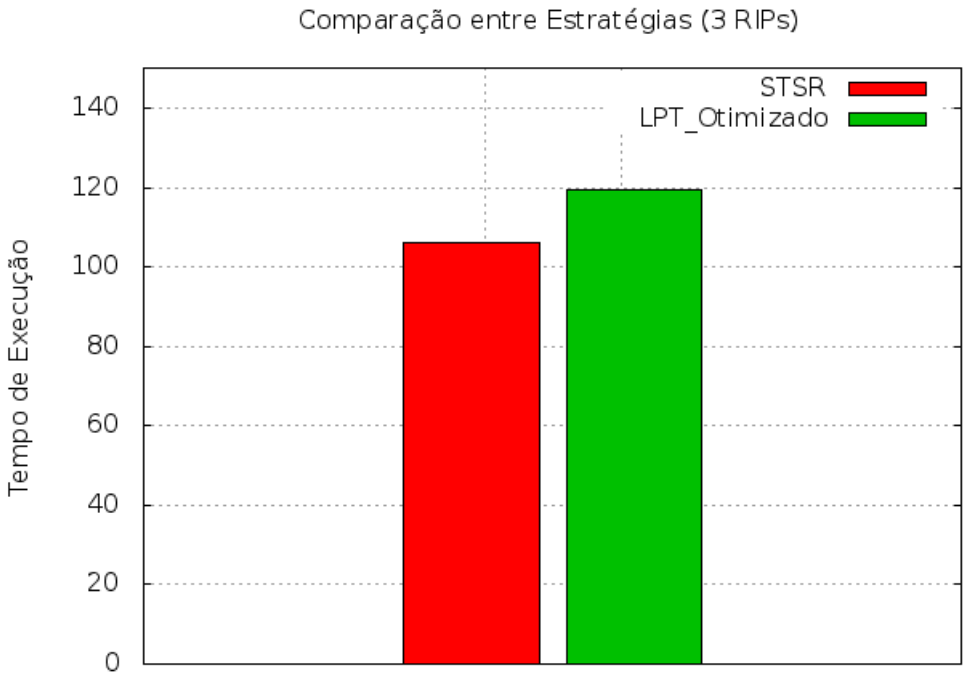

Figura 6.5: Gráfico de comparação entre a estratégia Sem Transparência e Sem Reusabilidade (STSR) e LPT Otimizado com uma distribuição de 3 RIPs.

7. AVALIAÇÃO DO RAJ

Este capítulo aborda o ambiente de teste e os casos de testes utilizados para executar o Roteador Adaptativo de *Jobs*. Também, demonstra-se os resultados obtidos com o uso das cinco estratégias desenvolvidas comparando-as com o algoritmo que obteve o melhor desempenho em [19]. Por fim, tem-se a análise do ganho médio. Os resultados foram obtidos através de uma média de 30 execuções com o uso de um RIP *open-source*, o *ImageMagick converter*. Nos testes, os documentos PDFs foram rasterizados utilizado-se 40 DPI (*Dots per Inchs*) de resolução. Cabe ressaltar que o conjunto de testes utilizado não é exaustivo, ou seja, não conseguiu abranger todos os tipos possíveis de documentos.

7.1 Ambiente de Teste

Será descrito nesta seção o ambiente de *hardware* e *software* utilizado para executar e implementar as estratégias e o Roteador Adaptativo de *Jobs*. Além disso, serão apresentados os *jobs* utilizados para realização dos testes.

7.1.1 Ambiente de *Hardware* e *Software*

Para se aproximar da realidade das PSPs, o ambiente de *hardware* que foi utilizado para realização dos testes é um agregado computacional constituído por 32 máquinas interligadas por meio de uma rede de alta velocidade *Gigabit Ethernet*. Cada máquina é composta por dois processadores AMD *Opteron* 246 2.0 GHz, 8 GB de memória principal e 1 TB de disco rígido. O sistema operacional destas máquinas é o Linux (Ubuntu 4.2.4-1ubuntu4) versão 2.6.24.

Para implementar as cinco estratégias previamente descritas, utilizou-se a linguagem Java na versão 1.5 mesmo não sendo usual na área de alto processamento. Porém, esta linguagem vem oferecendo melhorias permitindo assim sua utilização (cresce o número de pesquisadores da área que utilizam esta liguagem para suas implementações) [9]. Além disso, esta linguagem oferece portabilidade, extensibilidade e alto nível de abstração, além de oferecer compatibilidade com a biblioteca Java PDFBox, utilizada no PDF *Profiler*. Assim com na Seção 6.1.1, para a criação do cojunto de *jobs* utilizou-se a bilbioteca Java iText na versão 2.0.7.

Como a arquitetura descrita acima é de um multicomputador, a comunicação entre os processos é feita através da troca de mensagens. Assim, utilizou-se a biblioteca MPJ-*Express* [18]. Esta biblioteca é uma implementação do padrão MPI *(Message Passing Interface)* [28] com o uso da linguagem Java.

7.1.2 Casos de Testes

Para avaliar o RAJ utilizando as cinco estratégias desenvolvidas, um conjunto de 75 *jobs* com características diferentes foi criado. A Tabela 7.1 mostra as características presentes nos *jobs* criados como o número de documentos, número de total de páginas, o número total de imagens e o número total de páginas com textos. Estes *jobs* estão divididos em cinco grupos: PDFs com transparências (Tabela 7.1a), PDFs com reusabilidade (Tabela 7.1b), PDFs com mais transparência do que reusabilidade (Tabela 7.1c), PDFs com mais reusabilidade do que transparência (Tabela 7.1d) e PDFs sem a presença de transparência e de reusabilidade (Tabela 7.1e).

Tabela 7.1: *Características dos jobs utilizados no RAJ.*

[a]

Job	Documentos	Páginas	Imagens	Páginas com Textos
Transp 1	1	80	20	80
Transp 2	1	100	40	100
Transp 3	1	90	30	90
Transp 4	1	85	30	85
Transp 5	1	90	25	90
Transp 6	1	120	40	80
Transp 7	1	100	38	100
Transp 8	1	80	35	90
Transp 9	1	85	40	85
Transp 10	1	120	36	90
Transp 11	1	125	48	80
Transp 12	1	140	46	100
Transp 13	1	110	42	90
Transp 14	1	95	40	85
Transp 15	1	130	45	90

[b]

Job	Documentos	Páginas	Imagens	Páginas com Textos
Reusab 1	20	100	100	100
Reusab 2	16	96	96	96
Reusab 3	18	126	126	96
Reusab 4	14	140	140	90
Reusab 5	10	100	70	96
Reusab 6	12	96	72	100
Reusab 7	14	168	140	96
Reusab 8	384	96	384	96
Reusab 9	360	90	360	90
Reusab 10	512	160	512	98
Reusab 11	16	96	96	98
Reusab 12	18	90	90	98
Reusab 13	14	98	98	98
Reusab 14	15	135	135	98
Reusab 15	22	220	220	98

[c]

Job	Documentos	Páginas	Imagens	Páginas com Textos
MaisTMenosR 1	1	100	38	100
MaisTMenosR 2	1	80	31	80
MaisTMenosR 3	1	90	42	90
MaisTMenosR 4	1	120	40	120
MaisTMenosR 5	1	100	36	100
MaisTMenosR 6	1	130	41	130
MaisTMenosR 7	1	85	37	85
MaisTMenosR 8	1	90	43	90
MaisTMenosR 9	1	110	38	110
MaisTMenosR 10	1	120	48	120
MaisTMenosR 11	1	80	38	80
MaisTMenosR 12	1	90	37	90
MaisTMenosR 13	1	130	37	130
MaisTMenosR 14	1	110	40	110
MaisTMenosR 15	1	85	37	85

[d]

Job	Documentos	Páginas	Imagens	Páginas com Textos
MenosTMaisR 1	1	100	34	100
MenosTMaisR 2	1	80	36	80
MenosTMaisR 3	1	90	30	90
MenosTMaisR 4	1	120	30	120
MenosTMaisR 5	1	110	34	110
MenosTMaisR 6	1	85	34	85
MenosTMaisR 7	1	110	36	110
MenosTMaisR 8	1	116	30	116
MenosTMaisR 9	1	130	30	130
MenosTMaisR 10	1	95	34	95
MenosTMaisR 11	1	80	34	80
MenosTMaisR 12	1	92	36	92
MenosTMaisR 13	1	98	30	98
MenosTMaisR 14	1	115	30	115
MenosTMaisR 15	1	100	34	100

[e]

Job	Documentos	Páginas	Imagens	Páginas com Textos
SemTSemR 1	1	100	20	100
SemTSemR 2	1	120	30	120
SemTSemR 3	1	100	25	100
SemTSemR 4	1	80	20	80
SemTSemR 5	1	60	21	60
SemTSemR 6	1	120	35	120
SemTSemR 7	1	95	26	95
SemTSemR 8	1	130	35	130
SemTSemR 9	1	90	29	90
SemTSemR 10	1	85	22	85
SemTSemR 11	1	100	34	100
SemTSemR 12	1	80	30	80
SemTSemR 13	1	70	27	70
SemTSemR 14	1	110	36	110
SemTSemR 15	1	115	40	115

Filas de *Jobs*

Para a execução do RAJ utilizando as cinco estratégias desenvolvidas, seis diferentes configurações de fila foram criadas. As configurações destas filas foram direcionadas para tratar das cinco estratégias criadas. A seguir, tem-se a descrição de cada fila:

- Fila 1: contém 15 documentos PDF com transparência, na ordem apresentada na Tabela 7.1a;
- Fila 2: contém 15 documentos PDF com reusabilidade, na ordem apresentada na Tabela 7.1b;
- Fila 3: contém 15 documentos PDF com mais transparência do que reusabilidade na ordem apresentados na Tabela 7.1c;
- Fila 4: contém 15 documentos PDF com mais reusabilidade do que transparência apresentados na Tabela 7.1d;
- Fila 5: contém 15 documentos PDF sem transparência e sem reusabilidade apresentados na Tabela 7.1e;
- Fila 6: contém 15 documentos PDF, divididos em 5 grupos com características diferentes, sendo eles (nesta ordem):
  - 3 com transparência (documentos Transp 1, Transp 2 e Transp 3 da Tabela 7.1a);
  - 3 com reusabilidade (documentos Reusab 1, Reusab 2 e Reusab 3 da Tabela 7.1b);
  - 3 com mais transparência do que reusabilidade (documentos MaisTMenosR 1, MaisTMenosR 2 e MaisTMenosR 3 da Tabela 7.1c);
  - 3 com mais reusabilidade do que transparência (documentos MenosTMaisR 1, MenosTMaisR 2 e MenosTMaisR 3 da Tabela 7.1d);
  - 3 sem transparência e sem reusabilidade (documentos SemTSemR 1, SemTSemR 2 e SemTSemR 3 da Tabela 7.1e);

7.2 Análise dos Resultados do RAJ

Esta seção apresenta os resultados obtidos com o uso do RAJ comparando-o com o LPT Otimizado. Neste sentido, 30 execuções foram realizadas sobre cada uma das filas descritas anteriormente. Destes resultados, foram retirados aqueles com maior e menor tempo para cada conjunto de testes, obtendo uma média dos outros 28 valores. Para cada execução das filas, 1 processo sempre será o escalonador, que tem como objetivo realizar a distribuição dos fragmentos. Com isto, para a existência de RIPs em paralelo o número de processos não deverá ser menor do que 3, ou seja, 1 escalonador e 2 RIPs. Para a execução dos testes variou-se o número de processos de 3 a 19. A seguir tem-se o desempenho de cada uma das estratégias.

7.2.1 Escalabilidade e Comportamento

Esta seção apresenta uma análise comparativa dos dois algoritmos para as seis filas visando identificar flutuações do comportamento e tendências na sua escalabilidade.

Fila somente com transparência (Fila 1)

A Figura 7.1, apresenta o gráfico com o desempenho do RAJ utilizando a estratégia Transparência comparada com o LPT Otimizado. Como pode-se verificar, o comportamento da curva do LPT Otimizado apresenta algumas flutuações nos tempos obtidos (com aumentos e diminuições nas medidas) gerando situações imprevisíveis. Para compreender melhor estas flutuações, deve-se analisar o funcionamento desta estratégia mais a fundo. Neste contexto, sabe-se que esta estratégia quebra os *jobs* em fragmentos de acordo com o número de RIPs. Porém, estes fragmentos não necessariamente têm seu custo calculado (conforme descrito na Seção 4.4), podendo acarretar um desbalanceamento das tarefas. Já a estratégia Transparência além de amenizar estas flutuações, obteve em média os melhores resultados (ganhando em 13 processos). Esta estratégia obteve o melhor desempenho absoluto (obtendo o menor tempo de processamento que foi de 118.4 segundos). A Tabela 7.2 contêm os valores de processamento de cada processo para as duas estratégias utilizadas.

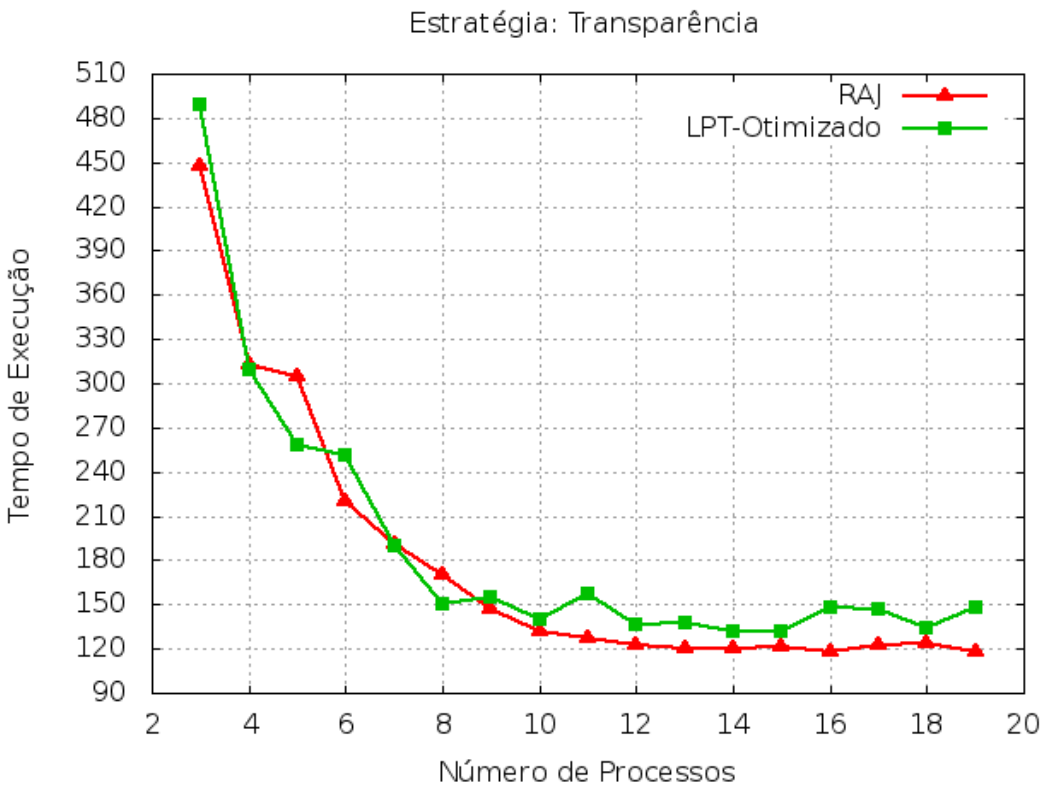

Figura 7.1: Gráfico de comparação entre a estratégia Transparência e o LPT Otimizado para uma distribuição de 3 RIPs.

Tabela 7.2: *Tempos de processamento (em segundos) na comparação da estratégia Transparência e LPT Otimizado.*

	3	4	5	6	7	8	9	10	11
RAJ	447.6	312.4	305.2	220.2	190.8	170.6	146.6	131.8	126.6
LPT-Otimizado	488.6	308.8	258.6	250.8	189.4	150.8	154.8	139.6	157.4
	12	13	14	15	16	17	18	19	
RAJ	122.4	120.2	119.8	121.2	117.6	122.8	123.8	118.4	
LPT-Otimizado	136.8	137.2	131.6	131.8	148.2	147.2	133.6	148.4	

Fila somente com reusabilidade (Fila 2)

A Figura 7.2 apresenta o gráfico com o desempenho do RAJ utilizando a estratégia Reusabilidade comparada com o LPT Otimizado. Pode-se verificar, o comportamento da curva do LPT Otimizado apresentou novamente flutuações nos tempos obtidos. Porém ele demostrou melhor desempenho do que a estratégia Reusabilidade, ganhando em 9 ocasiões (processos 4, 5, 6, 7, 9, 10, 12, 13 e 19). O ganho do LPT Otimizado em relação a estratégia Reusabilidade pode ser explicado pelo tamanho do grão escolhido. Eventualmente, o ganho com o agrupamento das páginas com imagens reusáveis em blocos pode ser pequeno, não compensando esta distribuição para os RIPs. Entretanto, a estratégia de Reusabilidade continua amenizando as flutuações apresentadas no LPT OTimizado. A Tabela 7.3 contêm os valores de processamento de cada processo para as duas estratégias utilizadas.

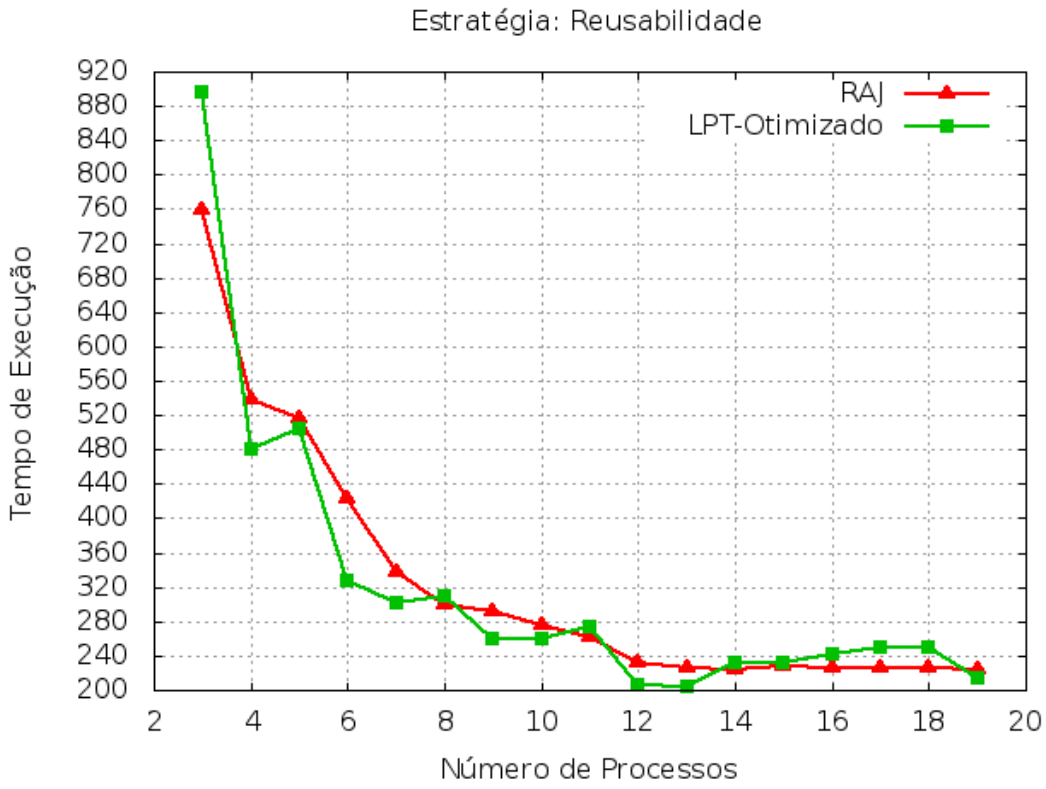

Figura 7.2: Gráfico de comparação entre a estratégia Reusabilidade e o LPT Otimizado para uma distribuição de 3 RIPs.

Tabela 7.3: *Tempos de processamento (em segundos) da estratégia Reusabilidade e do LPT Otimizado.*

	3	4	5	6	7	8	9	10	11
RAJ	758.4	537.8	515.4	422.2	337.6	300.2	292.4	274.8	261.2
LPT-Otimizado	896.8	480.6	504.8	328.2	301.4	310.2	260.6	259.6	273.6
	12	13	14	15	16	17	18	19	
RAJ	232.4	226.6	223.8	227.2	225.8	225.4	226.8	224.4	
LPT-Otimizado	205.6	204.4	231.6	232.2	241.4	250.2	249.8	213.6	

Fila com mais transparência do que reusabilidade (Fila 3)

A Figura 7.3 apresenta o gráfico com o desempenho do RAJ utilizando a estratégia Mais Transparência e Menos Reusabilidade comparada com o LPT Otimizado. Como pode-se verificar, o comportamento da curva do LPT Otimizado também apresenta algumas flutuações nos tempos obtidos, assim como na comparação com a estratégia Transparência e na Reusabilidade. Já a estratégia Mais Transparência e Menos Reusabilidade mesmo perdendo para o LPT Otimizado nos processos 4 e 5, em média ela obteve os melhores resultados (ganhando em 15 processos). Também foi possível notar como na comparação com a estratégia Transparência, que esta estratégia obteve o melhor desempenho absoluto (obtendo um menor tempo de 118.4 segundos). A Tabela 7.4 contêm os valores de processamento de cada processo para as duas estratégias utilizadas.

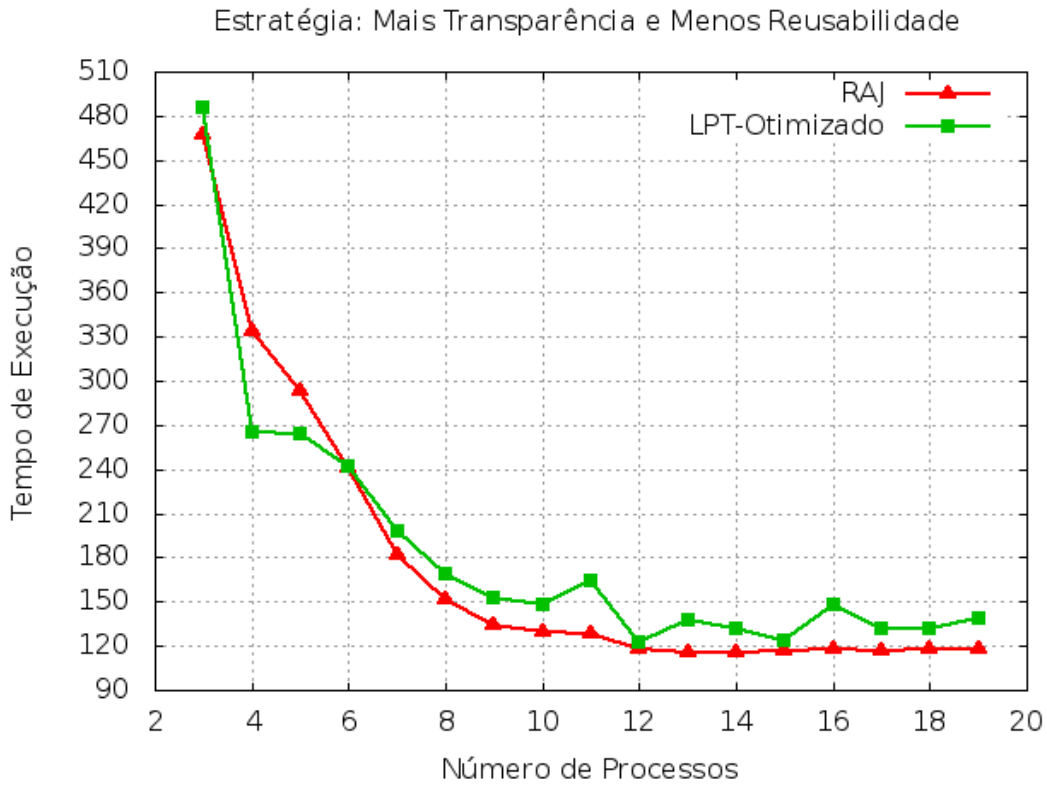

Figura 7.3: Gráfico de comparação entre a estratégia Mais Transparência e Menos Reusabilidade e o LPT Otimizado para uma distribuição de 3 RIPs.

Tabela 7.4: *Tempos de processamento (em segundos) da estratégia Mais Transparência e Menos Reusabilidade e do LPT Otimizado.*

	3	4	5	6	7	8	9	10	11
RAJ	467.4	333.8	292.8	240.4	181.8	151.6	134.2	129.6	128.4
LPT-Otimizado	485.4	265.6	264.6	241.6	197.8	169.4	153.2	148.4	164.2
	12	13	14	15	16	17	18	19	
RAJ	118.4	115.8	115.6	116.8	118.4	116.8	117.6	118.4	
LPT-Otimizado	122.6	137.8	132.2	123.4	147.8	131.8	131.6	139.2	

Fila com mais reusabilidade do que transparência (Fila 4)

A Figura 7.4 apresenta o gráfico com o desempenho do RAJ utilizando a estratégia Menos Transparência e Mais Reusabilidade comparada com o LPT Otimizado. Pode-se verificar que o comportamento da curva do LPT Otimizado também apresenta flutuações nos tempos obtidos. A estratégia Menos Transparência e Mais Reusabilidade mesmo perdendo em 4 ocasiões (processos 4, 5, 6 e 7) em média apresentou o melhor desempenho (13 ocasiões). Esta estratégia obteve o melhor ganho absoluto, com o tempo menor de 113,6 segundos. A Tabela 7.5 contêm os valores de processamento de cada processo para as duas estratégias utilizadas.

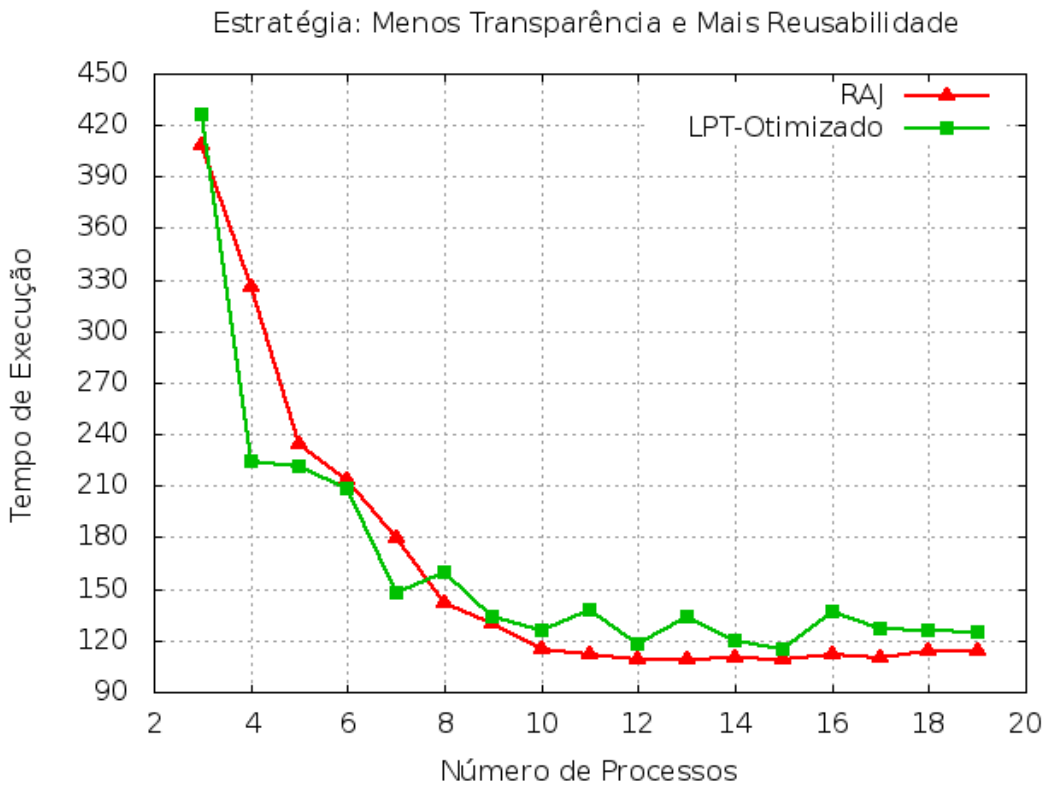

Figura 7.4: Gráfico de comparação entre a estratégia Menos Transparência e Mais Reusabilidade e o LPT Otimizado para uma distribuição de 3 RIPs.

Tabela 7.5: *Tempos de processamento (em segundos) da estratégia Menos Transparência e Mais Reusabilidade e do LPT Otimizado.*

	3	4	5	6	7	8	9	10	11
RAJ	408.4	325.4	234.6	213.6	179.8	141.6	129.8	114.6	111.8
LPT-Otimizado	425.8	224.2	221.4	208.6	147.8	159.2	133.4	125.4	137.4
	12	13	14	15	16	17	18	19	
RAJ	108.4	108.8	109.8	108.8	112.2	109.4	113.8	113.6	
LPT-Otimizado	117.4	134.2	120.2	115.2	136.6	127.2	126.2	124.4	

Fila sem transparência e sem reusabilidade (Fila 5)

A Figura 7.5 apresenta o gráfico com o desempenho do RAJ utilizando a estratégia Sem Transparência e Sem Reusabilidade comparada com o LPT Otimizado. Pode-se verificar que o comportamento da curva do LPT Otimizado novamente apresentou flutuações nos tempos obtidos. A estratégia Sem Transparência e Sem Reusabilidade perdeu em 2 ocasiões (processos 4 e 8) porém em média apresentou o melhor desempenho (15 ocasiões). Esta estratégia obteve também o melhor ganho absoluto, com o tempo menor de 107.2 segundos. A Tabela 7.6 contêm os valores de processamento de cada processo para as duas estratégias utilizadas.

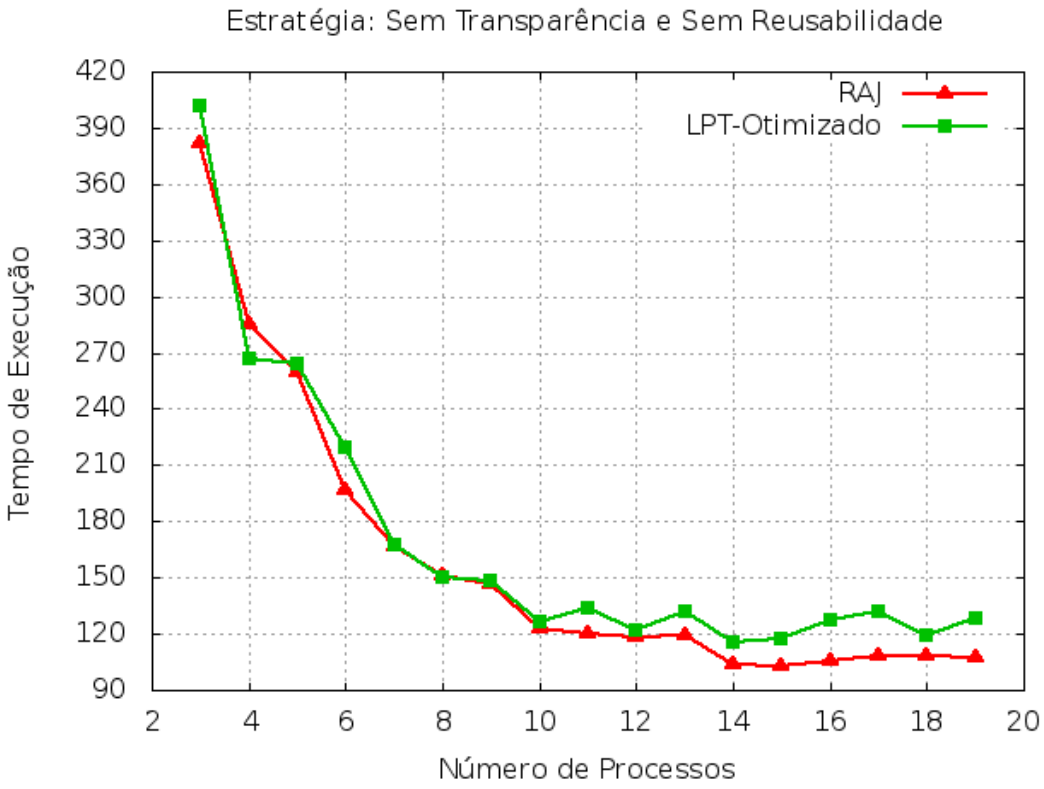

Figura 7.5: Gráfico de comparação entre a estratégia Sem Transparência e Sem Reusabilidade e o LPT Otimizado para uma distribuição de 3 RIPs.

Tabela 7.6: *Tempos de processamento (em segundos) da estratégia Sem Transparência e Sem Reusabilidade e do LPT Otimizado.*

	3	4	5	6	7	8	9	10	11
RAJ	381.8	285.2	259.8	196.6	166.6	150.8	146.2	122.4	120.2
LPT-Otimizado	401.8	266.4	263.8	219.4	167.6	150.2	148.2	126.4	133.8
	12	13	14	15	16	17	18	19	
RAJ	118.4	118.8	103.4	103.2	105.4	107.8	107.8	107.2	
LPT-Otimizado	121.6	132.2	115.4	117.8	127.4	132.2	119.2	128.2	

Fila com combinação de características (Fila 6)

A Figura 7.6 apresenta o gráfico com o desempenho do RAJ comparada com o LPT Otimizado utilizando a sexta fila (abordando assim, todas as cinco estratégias). Pode-se verificar, que o comportamento da curva do LPT Otimizado apresentou novamente flutuações nos tempos obtidos. Já o comportamento do RAJ amenizou estas flutuações e também pode-se observar o melhor ganho absoluto, com o tempo menor de 131,6 segundos. A Tabela 7.7 contém os valores de processamento do RAJ e do LPT Otimizado.

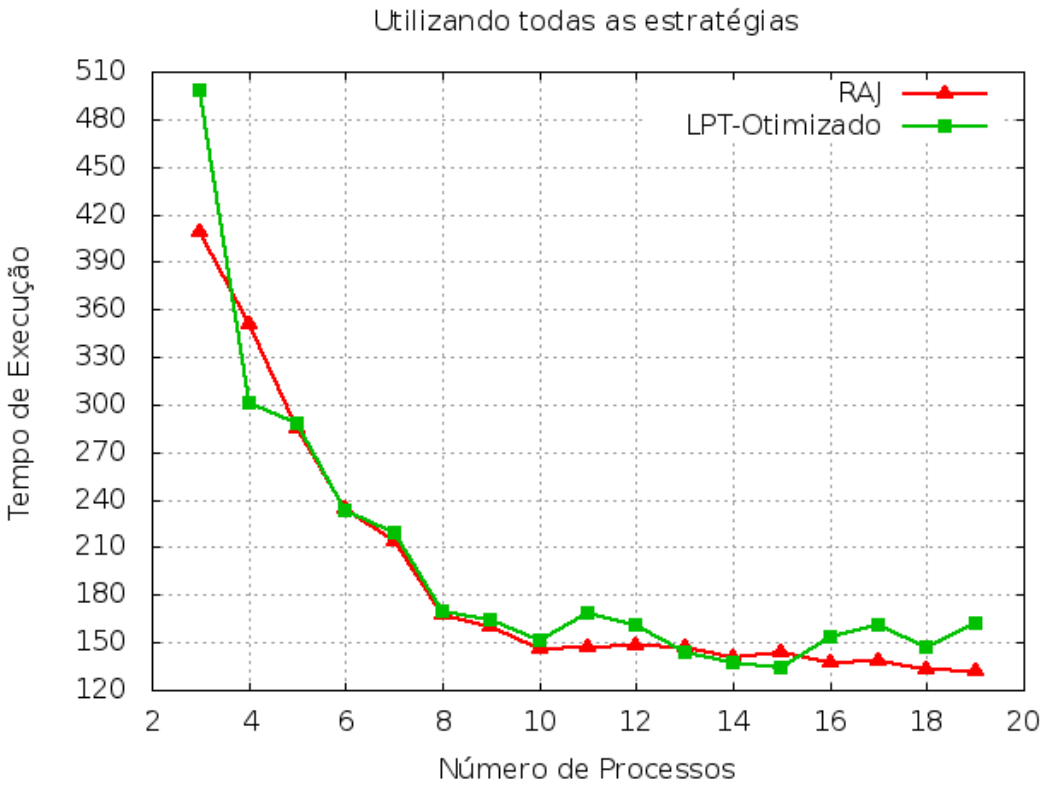

Figura 7.6: Gráfico de comparação entre o RAJ e o LPT Otimizado para uma distribuição de 3 RIPs utilizando a fila 6.

Tabela 7.7: *Tempos de processamento (em segundos) do RAJ e do LPT Otimizado utilizando a fila 6.*

	3	4	5	6	7	8	9	10	11
RAJ	408.6	350.2	285.2	233.8	213.2	167.6	159.6	145.4	147.2
LPT-Otimizado	498.4	300.6	288.2	232.8	218.6	169.4	164.2	150.8	168.6
	12	13	14	15	16	17	18	19	
RAJ	147.6	146.4	140.6	143.8	136.8	138.2	132.4	131.6	
LPT-Otimizado	160.8	143.4	137.2	133.6	153.4	161.2	146.8	162.2	

7.2.2 Análise do Ganho Médio

A Figura 7.7 apresenta os seis gráficos referentes ao percentual de diferenças entre as comparações do RAJ com o LPT Otimizado utilizando as seis filas descritas na Seção 7.1.2. Na Figura 7.7a, apenas os processos 4, 5, 7 e 8 obtiveram perda de 1.17%, 18.02%, 0.74% e 13.13% respectivamente. Porém, no geral a estratégia Transparência teve um desempenho melhor do que o LPT Otimizado, com uma média das diferenças percentuais igual a 7.22%, assim como na estratégia Menos Transparência e Mais Reusabilidade (Figura 7.7d), que obteve perda também em 4 processos (4, 5, 6 e 7), mas na maioria dos casos obteve um ganho melhor do que o LPT Otimizado com uma média das diferenças percentuais igual a 3.60%. Já as estratégias Mais Transparência e Menos Reusabilidade (Figura 7.7c) e Sem Transparência e Sem Reusabilidade (Figura 7.7e) obtiveram uma perda em apenas 2 processos (nos processos 4 e 5 e processos 4 e 8, respectivamente). Quando utilizada a combinação de características de documentos (Figura 7.7f), obteve-se uma perda em 5 processos (4, 6, 13, 14 e 15), mas assim como nas estratégias anteriores no geral obteve-se um ganho melhor do que o LPT Otimizado com uma média das diferenças percentuais igual a 4.38%. Entretanto, como se pode notar, apenas a estratégia Reusabilidade apresentou uma perda de maneira geral na média das diferenças percentuais (-2.79%). Como dito anteriormente, esta perda pode estar ligada ao tamanho escolhido do grão da tarefa. A Tabela 7.8 contém a média das diferenças percentuais de cada estratégia.

Tabela 7.8: *Média das diferenças percentuais.*

Filas	Média das diferenças percentuais
Transparência (Fila 1)	7.22%
Reusabilidade (Fila 2)	-2.79%
Mais Transparência e Menos Reusabilidade (Fila 3)	7.50%
Menos Transparência e Mais Reusabilidade (Fila 4)	3.60%
Sem Transparência e Sem Reusabilidade (Fila 5)	7.17%
Utilizando todas as estratégias (Fila 6)	4.38%

Figura 7.7: Gráfico de percentual de diferença.

Observando a Figura 7.7, nota-se que em todos os gráficos há uma perda significativa na configuração com 4 processos (onde 3 são RIPs). Esta perda pode ser ocasionada por um comportamento anormal do LPT OPtimizado. Com intuito de verificar a influência negativa dessa configuração no

desempenho médio de cada estratégia, foram feitas médias ignorando-se os valores da mesma. A seguir tem-se a Tabela 7.9 com a média das diferenças percentuais de cada estratégia retirando a configuração com 4 processos.

Tabela 7.9: *Média das diferenças percentuais retirando a configuração com 4 processos.*

Filas	Média das diferenças percentuais
Transparência (Fila 1)	7.59%
Reusabilidade (Fila 2)	-2.22%
Mais Transparência e Menos Reusabilidade (Fila 3)	9.57%
Menos Transparência e Mais Reusabilidade (Fila 4)	6.64%
Sem Transparência e Sem Reusabilidade (Fila 5)	8.06%
Utilizando todas as estratégias (Fila 6)	5.68%

Nota-se que ao retirar a configuração com 4 processos, todas as médias das diferenças percentuais tornaram-se melhores, apresentando um ganho. Como exemplo, a estratégia Menos Transparência e Mais Reusabilidade que tinha um ganho de 3.60% passou a ter uma ganho de 6.64%.

8. CONCLUSÃO

O trabalho realizado apresentou uma nova abordagem para a divisão dos *jobs*. Para tanto, uma modificação na ferramenta PDF *Profiler* com o intuito de obter o custo computacional de cada página foi realizada. Com isto, estratégias para a divisão dos *jobs* levando em consideração a características transparência e reusabilidade foram desenvolvidas. Também foi necessária uma modificação na ferramenta PDF *Splitter*, fazendo com que a divisão seja feita em páginas específicas. Um mecanismo automático foi desenvolvido para avaliar os *jobs* e definir qual das estratégias é adequada. Este mecanismo é chamado de Roteador Adaptativo de *Jobs* (RAJ).

Neste trabalho, cinco estratégias foram criadas, visando balancear a carga entre os RIPs. Cada uma delas divide a carga em fragmentos, de acordo com a principal característica do documento PDF. As estratégias criadas foram as seguintes:

- Transparência: divide a carga de trabalho (de forma balanceada) levando em conta as páginas com imagens transparentes para depois as páginas restantes do documento PDF (imagens opacas e/ou textos);
- Reusabilidade: divide a carga de trabalho (de forma balanceada) levando em conta os conjuntos com páginas reusáveis para depois balancear páginas restantes do documento PDF (imagens opacas e/ou textos);
- Mais Transparência e Menos Reusabilidade: divide a carga (de forma balanceada) de trabalho (de forma balanceada) levando em conta primeiramente as páginas com transparências. Após os conjuntos com páginas reusáveis para depois balancear páginas restantes do documento PDF;
- Menos Transparência e Mais Reusabilidade: divide a carga de trabalho (de forma balanceada) levando em conta primeiramente os conjuntos com páginas reusáveis. Após as páginas com imagens transparentes para depois balancear páginas restantes do documento PDF;
- Sem Transparência e Sem Reusabilidade: divide a carga de trabalho (de forma balanceada) levando em conta as páginas com imagens opacas e/ou texto.

Os resultados da análise do Roteador Adaptativo de *Jobs* no geral apresentaram um bom desempenho, apesar de um conjunto de testes não exaustivo. Nos gráficos dos tempos de execução, puderam-se perceber flutuações nos tempos do LPT Otimizado, o que não ocorreu nos tempos de execução do RAJ. Das seis filas criadas, apenas a fila que utiliza apenas a estratégia Reusabilidade obteve mais perda do que as outras, apresentando tempos de execução na maioria das vezes maiores do que o LPT Otimizado. As demais estratégias apresentaram diminuição nos tempos de execução na maioria dos casos.

Quando comparadas com o LPT Otimizado em nível de desempenho absoluto, entre as seis filas, cinco obtiveram o melhor desempenho absoluto. Além disso, os gráficos dos cenários criados (filas) mostraram que o RAJ apresenta uma tendência de redução do tempo de processamento quando mais processos forem adicionados, o que demonstra uma boa escalabilidade da implementação realizada, diferentemente do LPT Otimizado, que em boa parte dos casos não obteve ganho significativo de desempenho.

Os resultados da implentação foram satisfatórios para os objetivos do trabalho. Alguns pontos ainda podem ser melhorados, como, por exemplo, a realização de mais testes com mais conjuntos de *jobs* e utilizando filas com configurações e cenários diferentes.

8.1 Trabalhos Futuros

Durante a realização deste trabalho, identificaram-se alguns pontos de interesse para a continuação e aperfeiçoamento da pesquisa desenvolvida. Tratam-se de aspectos para melhoria das estratégias desenvolvidas e do Roteador Adaptativo de *jobs*.

- Portabilidade para máquinas multiprocessadas, pois a tendência é que os agregados de máquinas possuam cada vez mais máquinas multiprocessadas;
- Analisar outras características presentes nos documentos PDF como: *shadding pattern objects*, *inline objects* e *path objects*, para verificar o impacto destes elementos no tempo de rasterização;
- Aprimorar as estratégias desenvolvidas através da criação de novas métricas;
- Exploras novas estratégias aplicando diferentes características;
- Realizar um estudo para aprimorar o grão das tarefas para deixar o Roteador Adaptativo de *Jobs* otimizado.

Referências Bibliográficas

[1] Adobe Systems. "PostScript Language Reference Manual". Adobe Systems Incorporated, San Jose, USA, 1990.

[2] Adobe Systems. "PDF Reference". Adobe Systems Incorporated, San Jose, USA, 2003.

[3] Book, R. V. "Reducibility Among Combinatorial Problems", *The Journal of Symbolic Logic*, vol. 40–4, December 1975, pp. 618–619.

[4] Brucker, P. "Scheduling Algorithms". Secaucus, USA: Springer-Verlag New York, Inc., 2001.

[5] Coffman, E. G.; Garey, M. R.; Johnson, D. S. "An Application of Bin-packing to Multiprocessor Scheduling", *SIAM Journal of Computing*, vol. 7–7, February 1978, pp. 1–17.

[6] Davis, P.; deBronkart, D. "PPML (Personalized Print Markup Language): a New XML-based Industry Standard Print Language". In: XML Europe 2000, 2000, pp. 1–14.

[7] Extensible Markup Language. "XML", Extraído de http://wwww3org/XML Março de 2010.

[8] Garey, M. R.; Johnson, D. S. "Strong NP-Completeness Results: Motivation, Examples, and Implications", *Journal of ACM*, vol. 25–3, July 1978, pp. 499–508.

[9] Getov, V.; Hummel, S. F.; Mintchev, S. "High-performance Parallel Programming in Java: Exploiting Native Libraries", *Concurrency: Practice and Experience*, vol. 10–11, December 1998, pp. 863–872.

[10] Graham, R. L. "Bounds for Certain Multiprocessing Anomalies", *Bell System Technical Journal*, vol. 45–9, November 1966, pp. 1563–1581.

[11] Graham, R. L. "Bounds on Multiprocessing Timing Anomalies", *SIAM Journal on Applied Mathematics*, vol. 17–2, March 1969, pp. 416–429.

[12] ImageMagick Home Page. Extraído de http://www.imagemagick.org, Janeiro de 2010.

[13] iText Home Page. Extraído de http://itextpdf.com, Abril de 2010.

[14] Johnsonf, D. S.; Ullman, J. D.; Gareyi, M. R.; Grahamii, R. L. "Worst-case performance bounds for simple one-dimensional packing algorithms", 1974.

[15] Karlin, A. R.; Manasse, M. S.; Rudolph, L.; Sleator, D. D. "Competitive Snoopy Caching", *Algorithmica*, vol. 3–3, November 1988, pp. 79–119.

[16] Kruatrachue, B.; Lewis, T. "Grain Size Determination for Parallel Processing", *IEEE Software*, vol. 5–1, January 1988, pp. 23–32.

[17] Leung, J.; Kelly, L.; Anderson, J. H. "Handbook of Scheduling: Algorithms, Models, and Performance Analysis". Boca Raton, FL, USA: CRC Press, Inc., 2004.

[18] MPJ-Express Home Page. Extraído de http://mpj-express.org, Abril de 2010.

[19] Nunes, T. "Aplicando Estratégias de Escalonamento através da Análise do Perfil de Jobs para Ambientes de Impressão Distribuídos", Dissertação de Mestrado, Pontifícia Universidade do Rio Grande do Sul. Porto Alegre, Brasil, 2009.

[20] Nunes, T.; Fernandes, L. G.; Giannetti, F.; Cabeda, A.; Raeder, M.; Bedin, G. "An Improved Parallel XSL-FO Rendering for Personalized Documents". In: Euro PVM/MPI'07: Proceedings of the 14th European PVM/MPI Users Group Meeting. Recent Advances in Parallel Virtual Machine and Message Passing Interface (LNCS), 2007, pp. 56–63.

[21] Nunes, T.; Giannetti, F.; Fernandes, L. G.; Timmers, R.; Raeder, M.; Castro, M. "High Performance XSL-FO Rendering for Variable Data Printing". In: SAC'06: Proceedings of the 21st ACM Symposium on Applied Computing, 2006, pp. 811–817.

[22] Nunes, T.; Giannetti, F.; Kolberg, M.; Nemetz, R.; Cabeda, A.; Fernandes, L. G. "Job profiling in high performance printing". In: Proceedings of the 9th ACM symposium on Document engineering, 2009, pp. 109–118.

[23] Nunes, T.; Raeder, M.; Kolberg, M.; Fernandes, L. G.; Cabeda, A.; Giannetti, F. "High Performance Printing: Increasing Personalized Documents Rendering through PPML Jobs Profiling and Scheduling". In: Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 01, 2009, pp. 285–291.

[24] Pennebaker, W. B.; Mitchell, J. L. "JPEG Still Image Data Compression Standard". Norwell, USA: Kluwer Academic Publishers, 1992.

[25] Pinedo, M. "Scheduling: Theory, Algorithms and Systems". New York, USA: Prentice Hall, 2008.

[26] Purvis, L.; Harrington, S.; O'Sullivan, B.; Freuder, E. C. "Creating Personalized Documents: an Optimization Approach". In: Doc-Eng'03: Proceedings of the 2003 ACM Symposium on Document Engineering, 2003, pp. 68–77.

[27] Sleator, D. D.; Tarjan, R. E. "Amortized Efficiency of List Update and Paging Rules", *Communications of the ACM*, vol. 28–2, February 1985, pp. 202–208.

[28] Snir, M.; Otto, S.; Huss-Lederman, S.; Walker, D.; Dongarra, J. "MPI: The Complete Reference". Cambridge, USA: MIT Press, 1996.

[29] Zomaya, A. Y. H. "Parallel and Distributed Computing Handbook". New York, USA: McGraw-Hill, Incoraporated, 1996.