UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SISTEMA ESCALONADOR DE REQUISIÇÕES PARA SERVIDORES DE APLICAÇÕES DISTRIBUÍDAS

RODRIGO SIEWERDT

BLUMENAU
2006

2006/2-08

RODRIGO SIEWERDT

SISTEMA ESCALONADOR DE REQUISIÇÕES PARA SERVIDORES DE APLICAÇÕES DISTRIBUÍDAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação - Bacharelado.

Prof. Antônio Carlos Tavares , Especialista - Orientador

BLUMENAU
2006

2006/2-08

SISTEMA ESCALONADOR DE REQUISIÇÕES PARA SERVIDORES DE APLICAÇÕES DISTRIBUÍDAS

Por

RODRIGO SIEWERDT

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Antônio Carlos Tavares, Especialista – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB

Blumenau, 11 de dezembro de 2006.

Dedico este trabalho a minha família e todos os amigos que acreditaram na sua conclusão, especialmente aqueles que dedicaram alguns minutos de sua atenção, auxiliando nas decisões tomadas para o seu desenvolvimento.

AGRADECIMENTOS

À Deus, que guiou minhas atitudes e decisões durante toda minha vida.

À minha família, que esteve sempre ao meu lado em todas as horas, alegres ou difíceis, sem distinção.

Aos meus amigos, que tornaram minha caminhada mais amena e principalmente àqueles que me apoiaram e deram suporte quando necessário.

Ao meu orientador, Antônio Carlos Tavares, por ter acreditado na conclusão deste trabalho, por su disponibilidade e boa vontade, e aos professores que de alguma forma contribuíram para que houvesse melhorias no seu desenvolvimento, em especial à professora Dra. Fabiane Barreto Vavassori Benitti, que em ocasião alguma em que foi solicitada disse: “não tenho tempo”, mas sim respondeu: “claro, eu posso ajudar”.

“Não existe trabalho tão penoso que não possa ser proporcionado à força daquele que o faz, contanto que seja a razão e não a avareza que o regule.”

Montesquieu

RESUMO

Sistemas distribuídos estão cada vez mais presentes no cotidiano das empresas, pois visam a melhoria do desempenho global das aplicações utilizadas nas áreas de controle e operacionalização de processos. A possibilidade de utilização de vários computadores de pequeno porte para o processamento de informações, por si só, já representa um benefício considerável quando comparado à necessidade de servidores robustos e com custos elevados. Porém, um desafio que pode ser atribuído ao desenvolvimento de sistemas distribuídos, está relacionado com a correta distribuição de carga no processamento das informações. Algumas técnicas são utilizadas na tentativa de possibilitar a amenização desses problemas, sendo uma delas o escalonamento das tarefas. O objetivo deste trabalho é a construção de um ambiente de escalonamento de requisições para auxiliar na distribuição das tarefas num determinado conjunto de máquinas que possuam as aplicações servidoras instaladas. O sistema baseia-se na tecnologia DCOM e permite que aplicações escritas utilizando esta filosofia possam, desde que feito o tratamento necessário para a chamada das funções do escalonador, fazer uso das rotinas de escalonamento.

Palavras-chave: Sistemas distribuídos. Escalonamento. DCOM.

ABSTRACT

Distributed systems are every time more frequent in the daily of the companies, therefore they aim at the improvement of the global performance of the applications used in the control areas and in automation of processes. The possibility of use some small computers for the processing of information, by itself, already represents a considerable benefit when comparative to the necessity of robust servers with high costs. However, a challenge that can be attributed to the development of distributed systems is related with the correct load balance in the processing of the information. Some techniques are used in the attempt to make possible reduce of these problems, being the one of them the task scheduling. The objective of this work is the construction of an scheduling environment of solicitations to assist in the tasks distribution in one determined joint of machines with serving applications installed. The system is based on DCOM technology and allows that written applications using this philosophy can, since that made the necessary treatment for the call of the scheduler functions, to make use of the scheduling routines.

Key-words: Distributed Systems. Scheduling. DCOM.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

COM – *Component Object Model*

CPU – *Central Processor Unit*

DCE – *Distributed Computing Environment*

DCOM – *Distributed Component Object Model*

GUID – *Globally Unique Identifier*

HTTP – *Hyper Text Transfer Protocol*

I/O – *Input/Output*

IBM – International Business Machines Corporation

IP – *Internet Protocol*

OO – Orientação à Objetos

OSF – Open Software Foundation

RAM – *Random Access Memory*

RPC – *Remote Procedure Call*

UML – *Unified Modeling Language*

UC – *Use Case*

SUMÁRIO

1 INTRODUÇÃO

A constante busca por inovação tecnológica e o crescimento do setor de informática nas últimas décadas, com aplicativos cada vez mais complexos para atender necessidades do mercado, provocou o aumento da robustez nas operações executadas pelos sistemas de controle de informações. Por sua vez, a complexidade envolvida evidenciou a necessidade de computadores cada vez mais potentes, devido ao elevado processamento necessário para estas tarefas.

Para amenizar essa deficiência, surgiu o conceito de sistemas distribuídos, partilhando a execução das tarefas e reduzindo a sobrecarga nos servidores de aplicações. Além de colaborar para o aumento do desempenho, são também propriedades dos sistemas distribuídos maior flexibilidade e robustez. Flexibilidade diz respeito à adaptação às mudanças de ambiente sem perder funcionalidade e sem necessidade de alteração da estrutura do sistema. Assim devem permitir a adição ou retirada de componentes para atender novos requisitos. Por sua vez, a robustez indica o funcionamento do sistema após uma falha. Esta característica tem forte relação com os aspectos de redundância, natural nos sistemas distribuídos, observando-se que existe uma certa independência dos diversos componentes. Isto agrega um certo grau de confiabilidade à aplicação que utiliza essa estrutura, dizem Queiroz e Cunha (1994).

De acordo com Lages e Nogueira (1986), o conceito de sistemas de processamento distribuído é um dos campos de pesquisa e desenvolvimento de maior significado na aplicação de computadores. O interesse por sua utilização se deve ao rápido crescimento da tecnologia de circuitos integrados, que torna os mini e microcomputadores cada vez mais poderosos, utilizando processadores com capacidades extremamente maiores e com recursos antes inimagináveis, a um custo bastante acessível, tornando viável o processamento integrado de sistemas complexos de forma distribuída, levando-se em conta as necessidades cada vez mais sofisticadas e apuradas dos usuários.

Segundo Elasser, Monien e Preis (2000), um desafio na construção de sistemas distribuídos é promover o deslocamento da carga de servidores sobrecarregados para equipamentos que estejam sem ou com nível baixo de carga. Este contexto vem estimulando estudos sobre a otimização do balanceamento de carga para aplicações distribuídas em redes de computadores e estações de trabalho.

Nas abordagens tradicionais de estratégias de balanceamento de carga em servidores distribuídos, o processo é contínuo e iterativo e, geralmente, dividido em monitorar a carga

dos servidores por um determinado tempo, determinar a quantidade de carga que deve migrar para outros servidores baseando-se na observação efetuada e migrar a carga estabelecida.

Porém, existe um problema envolvendo esse processo, que diz respeito ao tempo utilizado nas observações. Caso seja pequeno demais, o custo de migração poderá ser muito alto e o balanceamento pode não surtir efeito. Por outro lado, se o tempo for muito grande, o desbalanceamento pode perdurar o bastante, chegando a afetar o desempenho global dos servidores. Sendo assim, a determinação do tempo é uma questão imprescindível e depende de características tanto do sistema quanto da aplicação.

Outra técnica utilizada para evitar a sobrecarga é descrita como escalonamento de processos. Pela ótica de Bodart et al (2004), essas atividades tem grande importância na utilização de sistemas distribuídos, sendo que o melhor desempenho e a otimização dos sistemas estão intimamente ligados com o uso adequado da capacidade de processamento oferecido pela estrutura instalada.

É com este intuito que o trabalho em questão está sendo proposto, buscando-se desenvolver uma nova ferramenta no tratamento do escalonamento dos processos solicitados, visando equilibrar as tarefas de processamento dos sistemas, possibilitando um melhor uso dos recursos disponibilizados.

Para aumentar o desempenho dos sistemas e reduzir os custos de obtenção e manutenção dos equipamentos, utiliza-se o conceito de processamento distribuído, onde ocorre a distribuição das operações de um sistema entre vários equipamentos. Utilizando esta tecnologia não existe a necessidade de servidores muito robustos para atender essa condição, porém as tarefas continuam igualmente rápidas e seguras. Para alcançar esse objetivo, entretanto, é necessário observar alguns detalhes e sanar alguns possíveis problemas através de uma aplicação externa, com finalidade de evitar a sobrecarga dos servidores e distribuir as requisições de operações advindas dos clientes, de forma transparente e homogênea, levando em consideração a disponibilidade de recursos dos equipamentos utilizados (servidores).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um escalonador de processos para dividir as tarefas e balancear a carga de processamento entre servidores.

Os objetivos específicos do trabalho são:

a) desenvolver um aplicativo escalonador baseado em processamento distribuído para comunicação entre objetos;
b) implementar uma aplicação cliente e servidor para comunicação com o componente de escalonamento, afim de que uma simulação do processo descrito seja possível.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos.

No primeiro capítulo tem-se a introdução, a justificativa do trabalho, o objetivo geral e objetivos específicos.

A fundamentação teórica sobre o tema abordado no trabalho está sendo exposta no segundo capítulo. Neste capítulo são expostos e tratados os assuntos relacionados com sistemas distribuídos, as técnicas utilizadas na otimização de desempenho desse tipo de sistema, escalonamento de processos, as classes principais de escalonadores, os tipos de algoritmos envolvidos e as políticas relacionadas, a tecnologia COM/DCOM de comunicação entre objetos e por fim os trabalhos correlatos.

No terceiro capítulo podem ser visualizados o projeto do sistema e as técnicas utilizadas para a implementação do trabalho, contendo um estudo sobre a operacionalização do sistema e os resultados obtidos com o uso desta ferramenta.

A conclusão e as sugestões para extensões do trabalho ou novas implementações estão descritas no quarto capítulo, que finaliza o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem a finalidade de apresentar de forma objetiva os conceitos de sistemas distribuídos, as vantagens da utilização desse tipo de sistema, as técnicas utilizadas para melhorar o desempenho e os estudos relacionados com a área de escalonamento de processos, os quais fundamentam o presente trabalho.

Ainda apresenta alguns conceitos sobre a tecnologia utilizada na elaboração e desenvolvimento da aplicação.

2.1 SISTEMAS DISTRIBUÍDOS

Segundo Strack (1984), após algum tempo pesquisando a evolução dos computadores durante as décadas passadas desde a concepção do Electronic Numerical Integrator and Calculator (Eniac), construído a base de válvulas em 1944, concluiu que a microeletrônica deu um grande salto na década de 70. Devido ao surgimento do microprocessador, que em uma pastilha de alguns milímetros quadrados apresenta as funcionalidades básicas de um computador, ocorreu uma rápida proliferação e popularização dos microcomputadores. Este processo teve ainda o acompanhamento de um considerável decréscimo, que aconteceu de forma contínua e progressiva na relação preço x *performance*.

Strack (1984), afirma ainda que as inovações na capacidade de processamento das rotinas de comunicação, que ocorreram a partir deste mesmo período, proporcionaram um grande desenvolvimento das redes de computadores. Com isto, foi adicionado o custo de processamento das atividades de telecomunicações ao processamento de dados. Com o passar do tempo essas atividades foram tornando-se uma parcela considerável nesse processo.

Uma das propriedades mais importante dos sistemas de processamento eletrônico de dados é o seu múltiplo uso, ou seja, sua utilização por vários usuários ao mesmo tempo e sendo assim, o seu projeto é desenvolvido como em qualquer ramo da engenharia, para atender certas especificações de *performance*. As metodologias e procedimentos de avaliação podem ser utilizados pelos projetistas na fase de construção, como também pelos usuários durante a utilização dos computadores, e o objetivo básico disso é otimizar o uso de todos ou,

pelo menos, dos recursos mais caros, com a finalidade de maximizar a quantidade de trabalho realizado durante o período de operação dos sistemas.

Neste contexto, no fim dos anos 70, surgiram os sistemas distribuídos integrando os recursos computacionais e de telecomunicações, de forma a reduzir os custos da transmissão de dados e colocar a capacidade de processamento junto ao usuário final da aplicação, além de utilizar de forma mais intensa o poder de processamento agora presente nos computadores pessoais, ou microcomputadores.

Pitanga (2003) descreve que os supercomputadores são sistemas que utilizam um elevado poder de processamento, sendo capazes de realizar muitas tarefas ao mesmo tempo, na tentativa de atender seus usuários da forma mais rápida possível. Muitos dos recursos contidos nos projetos dos antigos supercomputadores e de demais sistemas foram incorporados aos computadores pessoais de nossa atualidade, levando um moderno computador pessoal de mesa a oferecer mais poder de processamento do que um supercomputador de 15 anos atrás.

Os antigos computadores localizavam-se em grandes salas destinadas apenas ao seu uso, em que a programação era baseada em cartões perfurados. A computação dos anos 60 e 70 foi dominada por mainframes – grandes máquinas que poderiam suportar uma centena de usuários simultaneamente acessando o sistema através de terminais denominados “burros”.

Em contrapartida, o desenvolvimento da arquitetura de máquinas vetoriais significou que se poderia processar grandes quantidades de dados matemáticos muito mais rápido do que qualquer outro tipo de processador, embora as CPUs vetoriais pudessem ser mais lentas se utilizassem instruções mais complexas. Os modelos de supercomputadores vetoriais foram em seu tempo responsáveis pelo desenvolvimento dos sistemas mais rápidos do mundo, que contribuíram massivamente para as tecnologias posteriores, é o que diz Pitanga (2003).

Um elevado número de processadores pode ser agrupado em um supercomputador com processadores vetoriais, explica Pitanga (2003), mas está frequentemente limitado pela capacidade para se ter acesso a uma grande quantidade de memória compartilhada. Foi para superar esta limitação que surgiram os sistemas massivamente paralelos, que permitem a conexão entre processadores e sua memória local através de uma arquitetura de rede dedicada, aumentando a escalabilidade e limitando-se somente por altos custos.

De acordo com Berson (1996), o multiprocessamento vem se tornando uma ferramenta indispensável para aumentar a performance dos sistemas computacionais, possibilitando que os mesmos suportem aplicações cada vez mais complexas e robustas. De fato, sistemas de

multiprocessamento e paralelismo contribuem para que sejam alcançadas novas marcas de performance, escalabilidade e disponibilidade por parte das aplicações.

Geralmente, adicionar processadores numa máquina cria a possibilidade de executar diversas tarefas computacionais em paralelo, consequentemente aumentando o desempenho de um determinado programa. Em termos de custo, a curto prazo, a solução pode parecer ideal e demonstrar ganhos substanciais quando analisando-se um programa em separado, bem porque não exige custos de manutenção de sistemas para possibilitá-los executar em rede, de forma distribuída. Porém, comparando-se com a utilização de um sistema distribuído, onde as tarefas possam ser executadas em diversas máquinas diferentes, são somados ainda aos resultados, a escalabilidade e disponibilidade dos serviços.

Strack (1984) destaca que a tecnologia dos computadores, as formas operacionais, as alternativas de processamento e as próprias aplicações computacionais estão convergindo na importância dos sistemas distribuídos na realidade do processamento eletrônico de dados.

Mas Strack (1984) enfatiza que, entretanto, não existe uma definição de sistemas distribuídos com a qual todos os especialistas desta área concordem. Esta falta de uniformidade é, em parte, devido ao fato da distribuição ocorrer em qualquer nível, como: arquitetura interna das máquinas, disposição física, modo de interconexão, sistemas operacionais, controle, dados ou aplicações. Desta forma, uma definição segura e abrangente, deve levar em consideração todos ou boa parte desses fatores. Em alguns casos, para determinar se um sistema é ou não distribuído, estuda-se inclusive as dimensões do equipamento, controle e dados.

Amorim, Barbosa e Fernandes (1988), já diziam que o primeiro critério para classificação de um sistema distribuído está relacionado quanto a forma com que é feita a comunicação entre os processadores envolvidos no sistema. Tendo por base esse critério, uma divisão bem ampla pode ser feita entre sistemas distribuídos onde a comunicação é feita por meio de *broadcast*, e aqueles em que a comunicação se dá de forma ponto-a-ponto.

Quando a comunicação utiliza-se do *broadcast*, todos os processadores do sistema podem se comunicar diretamente entre si, através de um (ou vários) canais de comunicação direta, compartilhados por eles. Nestes sistemas, existe o problema claro de colisões quando dois ou mais processadores tentam utilizar o meio compartilhado simultaneamente.

Em contraste ao modelo apresentado anteriormente, existem ainda aqueles que utilizam a comunicação de forma ponto-a-ponto. Neste modelo, um processador mantém um canal de comunicação direto e exclusivo com outro processador ao qual deseja-se obter ou enviar informações. Sua representação pode ser feita através de grafos conexos em que cada

nó representa um processador distinto e cada arco entre dois nós representa um canal de comunicação de uso exclusivo dos dois processadores representados por aqueles nós. Esta exclusividade no uso dos diversos canais elimina o problema das colisões, mas causa por outro lado a necessidade de mensagens precisarem ser enviadas ao longo de rotas que passam por outros processadores que não sua origem e seu destino. A figura 1 representa o grafo aludido ao conceito de sistemas distribuídos ponto-a-ponto.

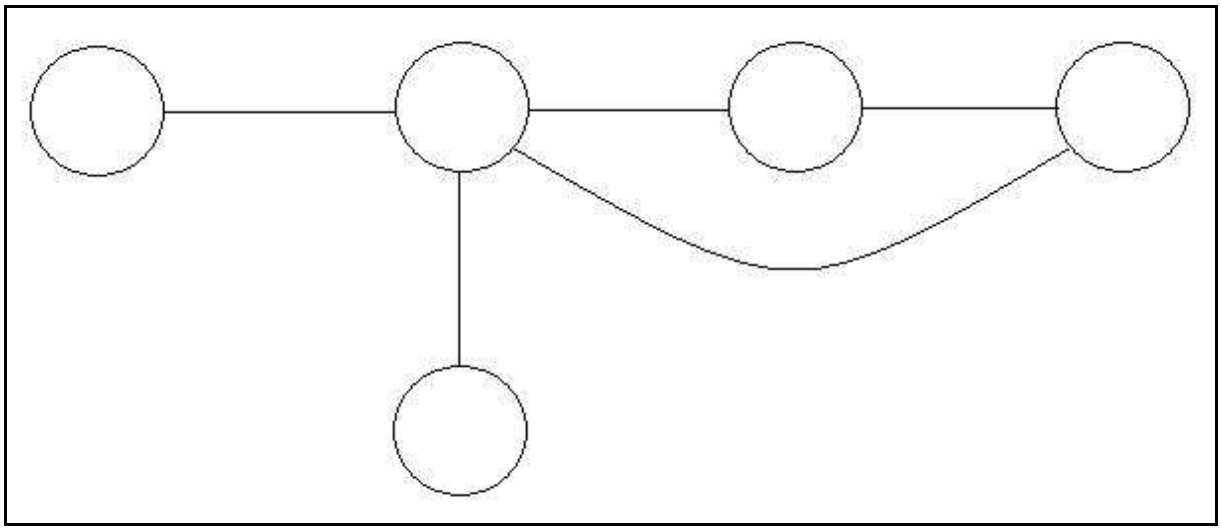

Figura 1 – Grafo de sistemas distribuídos

Definido de forma simples, um sistema distribuído é um conjunto de recursos de software e hardware relacionados trabalhando em conjunto para alcançar uma função comum. Múltiplas aplicações de gestão, serviços *web*, *sites* de comércio eletrônico e centrais de dados de empresas são exemplos destes sistemas (MICROSOFT, 2005).

Os sistemas distribuídos estão cada vez mais presentes no cotidiano dos profissionais de informática e o desenvolvimento de aplicações baseadas na utilização de objetos distribuídos constitui um campo de pesquisa em crescente desenvolvimento. De acordo com Queiroz e Cunha (1994), isto ocorre pelas facilidades que são oferecidas pelos ambientes computacionais típicos implantados atualmente, principalmente pelas redes de computadores encontradas em qualquer empresa, providas de vários microcomputadores independentes, ou seja, que possuem seus próprios processadores e memória física e que por esta razão, são ideais para a implementação de sistemas que tem o objetivo de trabalhar de forma distribuída.

A tendência é de que o crescimento seja progressivo, chegando a um ponto onde a distribuição dos componentes seja um requisito básico na construção da maioria dos sistemas de controle. Para Branco (2004), os sistemas computacionais distribuídos estão ganhando muita importância nas últimas décadas, sendo um dos conceitos de computação preferido

quando comparado à computação centralizada. O surgimento de microprocessadores cada vez mais potentes e com custo mais atrativo, aliado ao avanço na tecnologia de comunicação de dados, ocasionando a possibilidade do uso de redes de computadores de alta velocidade, foram os principais estimuladores do interesse pela utilização de sistemas distribuídos.

Apoiado em conceitos de orientação à objetos e utilizando essa capacidade de comunicação provida pelo uso das redes de computadores atuais, as aplicações de objetos distribuídos ganharam um impulso significativo com a expansão da internet, sendo que esse é um ambiente ainda mais favorável à sua utilização.

Um sistema distribuído é aquele que roda em um conjunto de máquinas sem memória compartilhada, máquinas que mesmo assim aparecem como um único computador para seus usuários, onde apenas o processamento e os dados são compartilhados entre as máquinas, porém a interface com o usuário é local e uma só.

Sistemas de compartilhamento de recursos, atualmente muito utilizados no desenvolvimento de sistemas integrados de controle em empresas, de acordo com Amorim, Barbosa e Fernandes (1988) podem ser genericamente representados por um conjunto de processos e recursos. Cada recurso existente é compartilhado entre os processos, porém a utilização de cada um deles irá depender da necessidade e da intenção de uso por parte dos processos que normalmente comunicam-se através da troca de mensagens solicitando a utilização.

Ainda no entendimento de Amorim, Barbosa e Fernandes (1988), qualquer computação distribuída sobre os recursos compartilhados deve obedecer às seguintes exigências:

a) exclusão mútua: processos vizinhos não podem utilizar concorrentemente um recurso que eles compartilham.
b) ausência de “*deadlock*”: deve ser sempre possível a execução de pelo menos um processo.
c) ausência de “*starvation*”: um processo que queira utilizar um recurso deve conseguir fazê-lo em tempo finito.

Apesar do grande interesse pelos sistemas de processamento distribuído, as técnicas de escalonamento de processos ainda são pouco utilizadas pelas empresas, apesar dos inúmeros benefícios que oferece. Segundo Branco (2004) isso deve-se ao grande desafio que é a definição de técnicas e métricas eficientes para distribuição da carga de processos entre os elementos que fazem parte dos recursos utilizados por um sistema distribuído.

2.2 TÉCNICAS DE OTIMIZAÇÃO DE DESEMPENHO EM SISTEMAS DISTRIBUÍDOS

Segundo Nogueira et al (2001), os sistemas distribuídos são de fundamental importância no desenvolvimento de aplicações de alto desempenho. No entanto, ainda muito se debate sobre políticas de gerenciamento para os recursos computacionais fisicamente dispersos existentes nesses tipos de sistemas. Um grande número de políticas de balanceamento de carga têm sido propostas nos últimos anos procurando resolver o problema da ociosidade ou da super-utilização das máquinas em um sistema distribuído. Tais políticas, mesmo quando existe uma aparente simplicidade no tocante às tomadas de decisões de controle, ou seja, empregado um reduzido número de parâmetros, não possuem um comportamento fácil de ser previsto.

Sob determinadas condições, o comportamento dessas políticas pode se tornar excessivamente instável, tomando sucessivas decisões equivocadas e, como conseqüência, degradando de forma considerável o desempenho do sistema.

Diversos estudos indicam a importância de se explorar adequadamente o potencial representado pelo uso de plataformas distribuídas. Minimizar o tempo de execução, minimizar os atrasos na comunicação, maximizar a utilização dos recursos, maximizar o *throughput* do sistema, entre outros, são alguns dos objetivos desses estudos. Para alcançar esses objetivos destaca-se a utilização de um melhor gerenciamento e também de uma melhor alocação dos recursos relacionados à carga computacional do sistema (BRANCO, 2004).

Em um sistema distribuído, as tarefas devem ser alocadas nos processadores de tal forma que se faça a utilização eficiente dos recursos do sistema, utilizando-se para isso três técnicas básicas de alocação do processador, descritas assim por Pitanga (2003):

a) atribuição de tarefas: tem como objetivo inicial a melhoria do desempenho de execução de um processo;
b) balanceamento de carga: possui como meta a distribuição da carga igualmente entre os nós da rede;
c) compartilhamento de carga: garante que nenhum nó fique ocioso enquanto algum processo espere para ser executado, ou seja, não permite que um processo fique na fila de espera enquanto houver um servidor disponível para atendê-lo.

Os algoritmos de alocação de processador devem levar em consideração requisitos como o tempo de resposta, a carga da rede, a sobrecarga e a maximização do uso do processador. A ausência desses requisitos ou o projeto falho de algum desses pontos, pode

comprometer sobremaneira a eficácia na utilização de sistemas distribuídos, acarretando perdas durante a execução das tarefas compreendidas nos sistemas.

Por esta razão, levando em consideração as características dos sistemas distribuídos, o escalonamento de processos é um item que merece muita atenção, já que é o responsável por esta correta distribuição de tarefas.

Para conhecer melhor o que é o escalonamento de processos, é interessante conhecer também um pouco sobre balanceamento de carga, já que, como destacado por Pitanga (2003), o balanceamento de carga entre servidores faz parte de uma solução abrangente em uma explosiva e crescente utilização das redes locais e da internet, provendo um aumento significativo na capacidade da rede, melhorando bastante o desempenho. Um consistente balanceamento de carga mostra-se hoje como parte integrante de todo projeto de hospedagem de sites e comércio eletrônico. Porém não se deve imaginar que isso esteja direcionado apenas para provedores de conteúdo e sim aproveitar todos os benefícios trazidos pela utilização destas técnicas e trazer para as empresas esse método de utilização da tecnologia, visando um atendimento mais rápido e eficiente dos clientes internos dessas empresas.

Spindola, Westphall e Koch (2004), destacam que no balanceamento de carga, algumas das principais metas são: evitar a utilização excessiva de um determinado recurso; equalizar a utilização dos recursos disponíveis; maximizar a utilização da capacidade do sistema e minimizar o tempo de execução das tarefas.

Pitanga (2003), acrescenta que quando não fazemos o balanceamento de carga entre servidores que aparentemente possuem a mesma capacidade de resposta a um cliente, ocorrem prejuízos e problemas aparecem, pois um ou mais servidores podem responder às requisições feitas e a comunicação fica prejudicada. Por isso é interessante colocar o elemento que fará o balanceamento de carga entre os servidores e os clientes e configurá-lo adequadamente para isso, fazendo com que um conjunto de múltiplos servidores de um lado pareça um único endereço para os clientes. Neste caso, esse elemento deverá ser um software dedicado a fazer todo esse gerenciamento, garantindo a passagem de pacotes e o balanceamento da carga de processamento entre as várias máquinas componentes desse aglomerado de servidores.

O escalonamento de processos utiliza a mesma premissa de garantir uma melhor utilização dos componentes pertencentes a um sistema distribuído, evitando a sobrecarga dos servidores e minimizando o tempo de resposta da aplicação.

2.3 VISÃO GERAL SOBRE ESCALONAMENTO DE PROCESSOS

O principal objetivo de um escalonador de processos é conhecer o grau de ociosidade dos servidores de aplicação que fazem parte do grupo que se queira escalonar. De acordo com Schlemer e Geyer (1998), saber o quanto da capacidade de processamento de cada máquina está sendo usado, é o ponto inicial para garantir um correto escalonamento da carga de processamento, já que em agregados de computadores existe uma grande probabilidade de um nodo ficar sobrecarregado enquanto outros ficam ociosos. Tal problema, conhecido como desbalanceamento de carga, degrada o desempenho do sistema como um todo, uma vez que uma distribuição de processos mais efetiva diminui significativamente o tempo de resposta computacional dos processos.

Pitanga (2003) acrescenta que o escalonamento de tarefas serve para encontrar a melhor solução para um problema, determinando o processo de melhor escolha das tarefas aos processadores, de tal modo que permita otimizar o desempenho do sistema. As estratégias de escalonamento são, na maioria das vezes, direcionadas na tentativa de se obter um menor tempo médio de respostas, minimizar os atrasos de comunicação e aumentar a utilização dos recursos disponíveis na rede.

Escalonamento de tarefas pode ser visto na computação como uma aplicação em diferentes cenários, sendo que uma das principais é o balanceamento de carga entre servidores. O balanceamento de carga pode ser definido como um caso especial de escalonamento de tarefas, que se baseia na questão de ser justo na distribuição de tarefas de acordo com os recursos de hardware do sistema, buscando sempre ser o mais vantajoso possível para seus usuários.

Algoritmos de escalonamento, segundo Pitanga (2003), são compostos por um método e uma política, os quais são projetados de forma a atuar o mais eficazmente possível sobre seus objetivos específicos, enfatizando que desempenho e eficiência são duas características importantes na avaliação de um sistema de escalonamento.

Seguindo a linha de raciocínio de Schlemer e Geyer (1998), os algoritmos de escalonamento de processos, são classificados utilizando-se algumas políticas, que são definidas a partir do momento que se possa visualizar as máquinas que estão com recursos disponíveis (aquelas que estão trabalhando aquém de sua capacidade) e as máquinas que estão trabalhando no limite de seus recursos, ou seja, estão sobrecarregadas.

Pitanga (2003) destaca que normalmente um escalonador é proposto seguindo algum critério de desempenho a ser melhorado, e o algoritmo empregado (também chamado de *scheduling*) é avaliado por sua complexidade de tempo, ou seja, a função que relaciona o tamanho de uma instância do problema ao tempo necessário para resolvê-lo. Diz ainda que se pode avaliar o desempenho de um escalonador observando o efeito causado sobre o seu ambiente de trabalho, através do seu comportamento na forma de como a política utilizada afeta os recursos e os utilizadores. E afirma também que nem sempre um ótimo algoritmo pode ser utilizado devido ao enorme tempo usado para obtenção dos resultados.

2.3.1 Classes principais de escalonadores

De acordo com Pitanga (2003), existem duas classes principais de escalonadores, que são chamados de determinísticos e não determinísticos.

Pinto e Dantas (2006) comprovam que a classificação das atividades de escalonamento global podem ser divididas em algoritmos estáticos e dinâmicos.

Em ambos os casos, as nomenclaturas têm o mesmo significado, portanto as duas divisões citadas serão descritas a seguir.

2.3.1.1 Algoritmos de escalonamento determinísticos

Para Pitanga (2003), um escalonamento determinístico, também conhecido como estático, configura que a decisão de escalonar as tarefas seja realizada em tempo de compilação, ou seja, todas as informações referentes ao conjunto de tarefas que compõem a aplicação são conhecidas antes de sua execução (número de processos, padrões de comunicação, granularidade dos processos e demais informações pertinentes ao escalonamento das tarefas). No método estático, um processo que inicia sua execução em um processador, roda no mesmo até a sua finalização e os nodos que fazem parte da rede que atenderá o sistema não se comunicam entre si para obtenção de informações, tampouco para troca de mensagens e redirecionamento de tarefas, evitando exaustivamente um custo elevado de comunicação e a sobrecarga da rede.

Pinto e Dantas (2006) acrescentam que a utilização de escalonamento estático necessita de um prévio conhecimento do comportamento e das dependências dos módulos de

um programa paralelo. Tal abordagem não leva em conta o estado atual do sistema, ou dos nodos pertencentes ao grupo. Como abordado anteriormente, também destacam que o escalonamento estático define o balanceamento das cargas antes da execução e ainda, que essa abordagem apresenta boa eficiência em ambientes homogêneos e processos cujo comportamento pode ser previsto em tempo de compilação.

Em um sistema onde a carga seja totalmente dinâmica, Pitanga (2003) resume que o escalonamento estático não é aconselhável, tendo um resultado imprevisível que em alguns casos pode estar muito aquém dos objetivos desejados com a utilização de algoritmos dessa abrangência, pois não se podem tomar decisões em tempo de execução do processo. Por esta razão, sistemas determinísticos fazem uso de algoritmos baseados em métodos aproximados e heurísticos que tendem a maximizar sua *performance*.

2.3.1.2 Algoritmos de escalonamento não-determinísticos

Já se tratando de algoritmos não-determinísticos, que podem também ser chamados de dinâmicos, a decisão sobre a distribuição dos processos entre os processadores é feita durante a execução, sendo praticamente desnecessário (ou pelo menos a relevância é bastante menor) conhecer as características dos processos antes que sejam executados, tornando o escalonador capaz de ajustar em função do comportamento do sistema durante a execução dos programas, daí o nome dinâmico.

Pinto e Dantas (2006) explanam que o escalonamento dinâmico é empregado nos casos em que as necessidades dos processos não são previamente conhecidas. Desta forma, o algoritmo de escalonamento deve consultar o estado do sistema constantemente, sendo que nenhuma decisão é tomada antes que o processo seja criado no ambiente.

Pitanga (2003) afirma que a alta complexidade desse método é um ponto negativo, no sentido de exigir informação em tempo real da carga do sistema e dos estados dos processos em cada nodo pertencente ao grupo, prejudicando muito a linha de comunicação que muitas vezes atinge um gargalo de sua capacidade, prejudicando em muito a eficiência deste modelo. Porém, por outro lado, a propensão de mudar seu estado durante a execução é um fator que contribui para um desempenho ótimo do processamento e em determinados casos incrementa a capacidade geral do sistema. Esse método é largamente empregado em ambientes multiusuários, como por exemplo os sistemas distribuídos e *clusters* computacionais.

2.3.1.2.1 Divisões dos algoritmos de escalonamento não-determinísticos

Ainda dentro da área de escalonamento dinâmico, temos os algoritmos do tipo não-distribuído, onde um processador é responsável pelas decisões de todo o sistema e os distribuídos, onde o algoritmo de escalonamento é executado por todos os nodos pertencentes ao grupo.

No escalonamento distribuído, cada estação possui as informações de carga dos demais nodos e, assim, decide para onde o processo deve migrar caso o processador esteja sobrecarregado. Neste modelo, ainda estudando os conceitos relatados por Pitanga (2003), o mesmo afirma que podem existir algoritmos cooperativos ou não-cooperativos, sendo que no primeiro caso as decisões individuais devem levar em consideração um objetivo comum para o sistema como um todo, tendo em vista uma política predefinida em que os processadores e processos colaboram entre si. Nos casos de algoritmos não-cooperativos, cada processador toma suas próprias decisões de forma independente dos demais processadores pertencentes ao grupo, sem haver preocupações sobre os resultados que isso porventura possa acarretar no sistema.

Na abordagem de escalonamento dinâmica, Pitanga (2003) destaca ainda a existência de algoritmos adaptativos, que são aqueles capazes de se adaptarem dinamicamente às mudanças no contexto do ambiente, em função dos estados anteriores e atuais do sistema (para que exista uma referência à tomada de decisões), isto é, em função do histórico do sistema. Para esclarecer o funcionamento desta técnica, podemos dizer que em resposta ao comportamento observado nesse histórico, o escalonador pode vir a recusar algum parâmetro ou diminuir a sua relevância se ele acreditar que o mesmo está fornecendo uma informação que é inconsistente para com o restante das entradas ou os dados fornecidos não contribuírem para a melhoria do desempenho global. O oposto, que seria um algoritmo não-adaptativo, não leva em consideração nenhuma informação de histórico do sistema em suas decisões, apresentando sempre o mesmo comportamento em situações semelhantes, pois não existe nenhum parâmetro para que se tome alguma decisão no sentido de mudar alguma das regras adotadas.

Subdivididos ainda em preemptivos e não-preemptivos, os algoritmos tomam mais uma vez duas direções distintas. Preemptivos, caracterizados pela migração dos processos, são aqueles em que uma tarefa é transferida de um nodo para o outro, mesmo que já tenha sido colocada em execução. Essa operação geralmente é custosa e difícil de ser realizada, uma

vez que será necessário reunir todas as informações sobre o estado da tarefa e transferi-la para o processador destino onde esta continuará sua execução. Já em sistemas não-preemptivos, as tarefas são escolhidas para distribuição antes de começarem a executar e, desse modo, não há transferência de processos entre os processadores. Uma vez iniciada a execução em um processador, a tarefa permanece nele até o fim de sua execução.

Sistemas distribuídos que fazem uso das técnicas de escalonamento dinâmico geralmente implementam a migração de processos, conforme explicado por Pinto e Dantas (2006). Quando esta ferramenta de migração é preemptiva (o processo pode ser transferido para outro nodo durante a sua execução), o processo deve ser migrado juntamente com seu contexto (espaço de endereçamento, *links* e arquivos abertos).

Após a verificação de todos esses modelos (e suas respectivas divisões) de escalonadores, é necessário ainda conhecer um pouco mais sobre outro quesito básico encontrado num algoritmo de escalonamento, pois como visto anteriormente, um escalonador é composto por métodos e políticas.

2.4 POLÍTICAS DE ESCALONAMENTO

Dentre as diversas políticas de escalonamento existentes, as que recebem maior ênfase no estudo deste trabalho, são as políticas de localização e de informação, já que o foco de estudo para elaboração do sistema proposto constante na proposta inicial toma por base uma união dessas duas.

Portanto, a seguir serão apresentadas, de forma resumida, as políticas mais conhecidas, aprofundando-se mais nas duas citadas anteriormente.

2.4.1 Política de Transferência

Quando falado sobre a política baseada na transferência, Pitanga (2003) destaca que se trata daquela que determina a viabilidade da realocação da tarefa e que a mesma está diretamente ligada à distribuição das cargas pelo sistema, ou melhor, é responsável por determinar se um nó encontra-se em situação adequada para receber mais tarefas para execução (receptor) ou para enviar tarefas que estejam gerando sobrecarga (transmissor).

Geralmente essa política baseia-se em valores limites de carga para determinar o estado em que o nó se encontra. Caso o nível de processamento esteja acima do limite, esse membro é considerado pela política de transferência como um nó transmissor, que precisa diminuir seu volume de carga. Em contrapartida, quando o processamento estiver abaixo dos valores limites estabelecidos, o membro torna-se (ou é eleito pela política de transferência) um receptor, estando apto a aceitar tarefas direcionadas para outras máquinas existentes no sistema.

Schlemer e Geyer (1998) destacam uma particularidade da política de transferência onde é preciso que se tome um cuidado especial. Trata-se da situação em que um doador (que tem sua carga maior que o limite) envia uma tarefa para um receptor (cuja carga é menor e encontra-se em condições de receber mais tarefas). A questão merecedora de tanta atenção supõe que este receptor, que tinha sua carga menor que o limite pré-estabelecido, ao receber esta nova tarefa passa a ser um doador, considerando que a inclusão da nova tarefa fez com que este limite fosse ultrapassado. Sendo ele um doador, as regras da política indicam que ele pode procurar um receptor para esta carga que causou o problema. Esta troca de tarefas pode ser feita várias vezes e até mesmo de forma indefinida, passando por vários membros que encontrem esta mesma situação.

Nestes casos fica claro que o algoritmo precisaria de algum tratamento, sendo uma alternativa incluir uma regra que considera um membro como doador não apenas aquele em que a carga for menor que o limite, mas também se continuará sendo menor ou igual após a passagem da tarefa em questão.

2.4.2 Política de Seleção

A política de seleção é responsável por selecionar quais os processos que devem participar da distribuição de cargas num ambiente de escalonamento de tarefas.

Logo que um nodo precisa enviar uma tarefa que o está onerando, levando em conta sua sobrecarga, ou quando o mesmo está apto a receber novas tarefas, caso esteja abaixo de seu limite de processamento, Schlemer e Geyer (1998) afirmam que este trabalho de decisão de qual será a tarefa escolhida fica por conta da política de seleção.

Pitanga (2003) ressalta que alguns fatores devem ser levados em consideração no momento dessa escolha, tais como:

a) a sobrecarga causada pela transferência deve ser mínima;

b) o processo a ser transferido deve requerer um tempo de processamento que compense a sobrecarga causada pela transferência;

c) o número de chamadas do sistema dependentes da localização deve ser mínimo. Nos casos em que a política de seleção considere também a distribuição das tarefas já em execução, cabem as regras de escalonamento realizadas de maneira preemptiva e não-preemptiva.

Quando for adotada uma regra de escalonamento preemptiva, uma tarefa que tenha sido escolhida para executar em determinado membro, mesmo após ter iniciado sua execução pode ser transferida para um outro processador, com o objetivo de retificar um balanceamento de carga que não tenha sido feito da forma ideal. Schlemer e Geyer (1998) destacam que este método pode ser muito oneroso, dados os custos de comunicação necessários para a sua transferência, tendo em vista que é necessário transferir todo o contexto da tarefa, como por exemplo os dados em memória. Geralmente opta-se pelas tarefas que tinham sido iniciadas mais recentemente, pois em teoria são as que devem possuir menores custos de transferência. Mas também existem casos em que a opção de escolha fica por conta das tarefas menores, sempre pensando nos custos advindos dessa escolha.

Já em escalonamentos que utilizam técnicas não-preemptivas, a transferência de tarefas ocorre apenas para os processos que ainda não tenham sido iniciados e, portanto, destacado por Schlemer e Geyer (1998), não necessitam de transferência de contexto. Pitanga (2003) enfatiza que nesse modelo as tarefas que tenham sido iniciadas em um determinado processador não podem ser transferidas e, dessa maneira, em um algoritmo sem preempção geralmente a tarefa selecionada para a troca de contexto é aquela criada mais recentemente e, conseqüentemente, a responsável pela sobrecarga do sistema.

2.4.3 Política de Localização

Consiste em escolher o membro desejado para receber a tarefa solicitada. Pitanga (2003) descreve que na política de localização determina-se o elemento de processamento ao qual uma tarefa deve ser associada, onde a seleção é baseada em uma carga mínima do processador ou no primeiro processador cuja carga seja menor que um certo limiar. Esta política tem por finalidade identificar para qual máquina do grupo o processo deve ser transferido.

Segundo Schlemer e Geyer (1998), o método mais utilizado nesta política para escolha da máquina é o randômico, ou seja, escolher uma máquina aleatoriamente para receber a tarefa. Mas existe um grande problema envolvido com este método, pois é possível que a máquina escolhida aleatoriamente esteja com um nível de carga muito elevado. Uma solução seria testar o nível de carga e ver se a mesma possui condições de receber a tarefa solicitada. Existe também um método onde se escolhe várias máquinas e dentre elas é feita uma votação para garantir que seja selecionada a que estiver em melhor condição de atender a requisição, porém, fica evidente a perda de *performance* devido ao excesso de comunicação.

2.4.3.1 Centralização e Descentralização

Há duas maneiras de se implementar a política de localização, conforme descrito por Pitanga (2003). São elas:

a) descentralizada: caso em que o nodo de destino pode ter sido escolhido basicamente por três maneiras: aleatoriamente (que é o mais utilizado, mas que pode gerar instabilidade no sistema), através de coleta de informações (onde o estado atual de todos os membros do grupo é analisado para escolher-se o melhor) ou através da análise dos vizinhos (caso em que são analisados os dados apenas dos nodos mais próximos).
b) centralizada: neste caso pode-se verificar a existência de um coordenador, ou seja, um dos membros do grupo conhece as informações de carga de todos os demais membros, e fica responsável por localizar o nodo de destino adequado ao compartilhamento da carga, tendo por base seu estado atual.

A quantidade de carga de um membro pode se basear no tamanho da fila de tarefas prontas (que já tenham sido executadas) deste nodo. Não é possível se ter uma medição exata do tamanho da tarefa devido ao fato de que as decisões são tomadas somente no tempo de balanceamento de carga.

2.4.3.2 Tipos de busca

Pitanga (2003) classifica as políticas de localização em três categorias distintas:

a) envio-iniciadas: são aquelas nas quais os remetentes procuram por receptores apropriados e que para cargas baixas até moderadas produzem um desempenho melhor, desde que os remetentes em potencial tenham facilidade de encontrar membros para onde possam transferir suas tarefas antes que fiquem sobrecarregados. Daí a importância de ser utilizado em ambientes com carga moderada, onde seja possível encontrar nodos sem muito acúmulo de processamento.
b) recepção-iniciadas: ao contrário da primeira, esta técnica consiste de receptores potenciais que procuram por remetentes. Neste tipo de busca, o nó sobrecarregado deve esperar por um contato vindo de um membro com pouca carga, e isso pode levar a uma perda de tempo significativa. Porém em ambientes com cargas muito altas, ou seja, quando as cargas de tarefas que chegam são elevadas, a busca recepção-iniciada produz um desempenho melhor do que as envio-iniciadas, desde que os membros menos sobrecarregados sejam capazes de encontrar outros membros sobrecarregados quase que imediatamente para que obtenha deles tarefas para serem processadas.
c) simetricamente-iniciadas: Esta técnica é uma junção das duas anteriores, portanto obtendo um desempenho mediano e sendo de melhor aplicação quando não se pode prever o nível de carga a qual o ambiente de escalonamento deverá suportar.

2.4.3.3 Estratégias de localização de transmissores e receptores

No algoritmo utilizado para aplicar a política de localização, a atividade de distribuição de carga é iniciada por um processador sobrecarregado (transmissor), tentando enviar uma tarefa para um nó ocioso. Pitanga (2003) afirma que existem três tipos de técnicas utilizadas para implementar esse algoritmo:

a) aleatória: nessa estratégia a tarefa é transferida para um processador selecionado aleatoriamente não fazendo uso de nenhuma informação de estado dos membros da rede. Dessa forma, uma transferência inútil pode ocorrer caso a tarefa seja encaminhada para um processador que já esteja sobrecarregado. Apesar desse grande problema relatado, esta estratégia é a mais comumente aplicada, conforme citado anteriormente e confirmado por Schlemer e Geyer (1998).

b) limite: essa estratégia tem como principal vantagem a capacidade de evitar uma distribuição inútil como a citada na utilização da estratégia anterior, pois inspeciona um processador (selecionado aleatoriamente) para determinar se a distribuição da tarefa excederá o seu limite de cargas. Em princípio, se não ultrapassar, a tarefa é transferida para o processador selecionado, o qual deve executar o processo sem se preocupar com seu estado quando a tarefa realmente chegar, pois a verificação foi efetuada anteriormente.

c) menor limite: nesse caso, um determinado número de processadores é selecionado aleatoriamente, sendo inspecionados para determinarem os tamanhos de suas filas de processos pendentes esperando para serem executados. O processador com a menor fila é selecionado como o processador destino da tarefa transferida.

2.4.4 Política de Informação

Nesta política de escalonamento, os membros participantes trocam informações entre si sobre o estado de cada um. Pitanga (2003) diz que a política de informação decide quando as informações sobre o estado de outros membros pertencentes ao sistema devem ser buscadas, de onde serão coletadas e quais informações devem ser trazidas.

Schlemer e Geyer (1998) descrevem que pode existir um armazenamento local em um membro específico (centralizada) que se encarrega de receber todas as mensagens de estado enviadas pelos outros membros, mas que isso causa um gargalo e sobrecarrega a máquina, não permitindo seu uso para recebimento de requisições das aplicações clientes. Existe também um outro método, que consiste no envio de um *broadcast* de cada membro para todos os outros (descentralizada) informando sobre seu estado, o que também causa um desperdício devido ao nível extremamente alto de comunicação entre as máquinas. O método mais tradicional funciona de modo que cada membro preocupa-se em manter apenas as informações sobre ele mesmo e os outros membros consultarem seu estado quando da troca de tarefas. Essa última é chamada de demanda, onde cada um preocupa-se apenas consigo mesmo e só procura obter informações do outro quando estiver sobrecarregado.

Pitanga (2003) subdivide em três tipos básicos as políticas de informação:

a) política dirigida por demanda: Nesta política, um membro do sistema coleta informações sobre o estado de outros membros somente quando estes tornam-se

também transmissores ou receptores disponíveis para iniciar e compartilhar cargas.

b) política periódica: realiza a captura de informações de maneira periódica. Em sua forma centralizada, de tempos em tempos, os nodos enviam informações do seu estado para um membro centralizador, que fica encarregado de administrar esses dados. Já quando aplicada a política de forma descentralizada, a informação de cada nó é distribuída entre as demais máquinas pertencentes ao grupo, que as utilizam na tomada de decisões.

c) política dirigida pela mudança de estado: com a utilização deste tipo de política, os membros difundem as informações sobre seus estados todas as vezes que estes mudam de uma forma significativa. Aqui também pode ser aplicada a teoria de centralizar as informações, caracterizando que as informações são enviadas para um único nodo coordenador. Nos casos em que se aplica uma política descentralizada, todos os membros comunicam-se entre si e as informações são enviadas a todos eles.

2.5 TECNOLOGIA *COM/DCOM*

Para implementação do trabalho apresentado, foi escolhida a tecnologia *Component Object Model / Distributed Component Object Model* (COM/DCOM) para implementação de componentes distribuídos. DCOM é um protocolo que permite que os componentes do software se comuniquem diretamente em uma rede, projetado para ser usado por diversos transportes de rede, incluindo protocolos de Internet como *Hyper Text Transfer Protocol* (HTTP) (MICROSOFT, 2003).

Eddon e Eddon (1998) lembram que começou a ser desenvolvido no início dos anos 80 e que naquele tempo o simples fato de conectar vários computadores utilizando uma rede local por si só já era um grande feito, que despendia trabalho e muito conhecimento. Como os computadores pessoais estavam em expansão, era muito importante que fosse desenvolvido algum tipo de sistema que conseguisse suportar algum tipo de comunicação distribuída, para fazer frente aos sistemas operacionais centralizados. Foi então que um grupo de indústrias de computadores se uniu cooperativamente traçando uma meta para a criação deste padrão. Esta decisão foi tomada pelo fato de que na indústria dos *mainframes*, padrões não existiam;

softwares eram escritos para computadores específicos e computadores IBM não tinham a mínima pretensão de comunicar-se com computadores fabricados por outras empresas.

Eddon e Eddon (1998) complementam que no final dos anos 80 já existiam várias parcerias de grupos industriais reunidos com o intuito de definir padrões computacionais que fossem acordados por todos e eles concordavam que isso seria importante. Um destes grupos, o Open Software Foundation (OSF), tornou-se um consórcio de empresas possuidor de um mandato para a definição dos padrões numa determinada área. Os membros da OSF decidiram direcionar seus conhecimentos e esforços para os caminhos da computação distribuída. A partir destes estudos nasceram as especificações conhecidas como Distributed Computing Environment (DCE). O ponto chave e o alvo principal da DCE eram prover um ambiente para criação de sistemas distribuídos. Quando esta meta foi atingida, a DCE começou a coletar uma gama de poderosas ferramentas integradas e serviços para suportar a criação de aplicações distribuídas, como uma maneira de fazer frente a todo o suporte oferecido pelos sistemas operacionais de ambiente centralizado.

Um dos recursos fornecidos pela OSF e pela DCE foi a especificação para comunicação entre computadores, conhecida com Remote Procedure Calls (RPCs), que oferecia um suporte para que aplicações rodando em diferentes computadores conseguissem se comunicar. DCOM utiliza RPCs para interconectar os computadores entre si e isso demonstra como o DCOM está envolvido com o RPC.

2.5.1 As três faces do COM

Eddon e Eddon (1998) acreditam que para entender o que o COM significa, faz-se necessário analisar três pontos, os quais serão demonstrados nas seções a seguir.

2.5.1.1 COM como uma especificação

O COM é uma especificação de software. Por isso é necessário compreender que ele é um documento que pode ser impresso e lido. Trata-se de uma série de documentos que especificam como funciona a construção de sistemas distribuídos e quais as diferenças que podem ser notadas na execução de programas desse tipo. Mas é importante lembrar, numa

análise completa, que o COM não trata apenas de documentos de especificação, mas também de uma gama muito abrangente de código fonte e ferramentas implementadas pela Microsoft.

2.5.1.2 COM como uma filosofia

COM é uma maneira de pensar sobre técnicas modernas de desenvolvimento de software. Sua especificação descreve um mundo onde cada aplicação é desenvolvida como um componente e, o mais importante, baseada nos conceitos de divisão por componentes. Uma grande quantidade de projetos de software ainda são implementados com a idéia de produzir um enorme programa único, contendo todos os recursos que um usuário possa precisar. O problema com este modelo de desenvolvimento de software é que as aplicações tornam-se cada vez mais frágeis e passíveis de erros conforme crescem. Além da dificuldade que se tem para entender e conhecer por completo uma aplicação contendo 100.000 linhas de código, por exemplo, e a quase impossibilidade quando se fala em 1.000.000 de linhas. Às vezes até mesmo uma modificação minúscula numa aplicação desse porte, requer um extensivo (e exaustivo) reteste de todo o sistema. E frequentemente o que parece ser uma inofensiva modificação num determinado setor do código, vem causar problemas em muitos outros locais.

Por isso na filosofia de COM, os dias das aplicações monolíticas estão acabando. No novo mundo criado por ele, desenvolvedores criam componentes compactos e bem definidos para trabalharem em conjunto. Esses componentes podem ser reutilizados em vários ambientes onde sejam necessários, tanto nas aplicações cliente como nas servidoras ou nas duas ao mesmo tempo. Inclusive por aplicações que tenham sido escritas em diferentes linguagens. O importante é que sejam utilizadas as regras definidas pelo COM, assim permitindo que a chamada do componente seja feita sem maiores problemas.

2.5.1.3 COM como um padrão de codificação

COM é o nome da especificação projetada pela Microsoft para definir o que a tecnologia básica de componentes significa e como um objeto COM deve ser utilizado.

A regras implementadas pela tecnologia COM pode ser considerada como a ligação entre os componentes desenvolvidos, habilitando que objetos de software sem nenhuma

relação entre si possam conectar-se e interagir de diversas formas. Para tornar sucinto, pode-se afirmar que o COM é um padrão de codificação de programas utilizado para permitir a integração de objetos.

Eddon e Eddon (1998) dizem que para entender melhor a razão para a existência da padronização oferecida pelo COM, é importante lembrar dos desafios propostos pelas indústrias de computadores que levaram a Microsoft a desenvolvê-la. O principal deles seria o de garantir que plataformas de hardware de diversos fabricantes pudessem executar um mesmo sistema. Então as discussões abordadas na especificação COM e que deveriam ser sanadas, foram:

a) aplicações modernas seriam grandes e complexas, consumindo muito tempo de desenvolvimento, manutenção complexa e cara, além de riscos sérios de comprometimento do software em casos de adição de funcionalidades;
b) as aplicações continuariam sendo desenvolvidas no estilo monolítico, contendo pré-definições de uma gama de recursos, não podendo ser adicionados, alterados ou removidos independentemente;
c) diferentes aplicações não seriam passíveis de integrações, ou seja, nenhum recurso ou funcionalidade de um determinado programa estaria disponível para outro programa;
d) os modelos de programação refletiam as definições dos fabricantes, sendo muito dependentes de onde um serviço estava sendo provido. Desta forma não era possível executá-los em outras plataformas, nem mesmo em outras máquinas pertencentes ao mesmo ambiente de rede, tampouco em diferentes sistemas operacionais.

Por estes motivos era necessário o surgimento de uma tecnologia que conseguisse efetuar a comunicação entre pequenos objetos de software, produzidos por fabricantes diferentes, porém seguindo uma mesma metodologia que permitisse essa transferência de informações.

Foi nesse contexto que surgiram os programas abrangendo um conjunto de funcionalidades para resolução de pequenas tarefas que são descritas e podem ser acessadas utilizando o conceito de interfaces. Nesta ótica, basta conhecer o padrão da interface para poder conectar-se a ela e utilizar todos os recursos oferecidos.

2.5.2 Evoluindo para componentes distribuídos

Na panorâmica de Horstmann e Kirtland (1997), o DCOM é uma extensão do COM para suportar comunicação entre objetos em computadores diferentes, utilizando redes locais e até mesmo a internet. Para criar componentes DCOM, são implementadas interfaces, que identificam as funções constantes num componente, e dentro do próprio componente são implementadas as funcionalidades desejadas. Esses componentes recebem *Globally Unique Identifiers* (GUIDs), ou identificação única global, que são utilizadas para identificar as classes particulares dos objetos, evitando colisões entre diferentes componentes.

Silva, Gomide e Petrillo (2003) afirmam que componentes fornecem um modelo padrão para empacotamento de serviços disponibilizados aos usuários (consumidores). Eles são um tipo de caixa-preta, pois todos os dados, tratamento de informações e detalhes de implementação permanecem completamente ocultos fora do escopo de desenvolvimento, sendo que os serviços são disponibilizados num modelo comum de implementação através de interfaces públicas associadas a eles. Para a construção de aplicativos servidores confiáveis, escaláveis e de alto desempenho, além de uma arquitetura baseada em componentes de várias camadas, é necessário uma infra-estrutura sofisticada, presente na categoria de software chamada de *middleware*. Este serve ainda como um "agrupador", permitindo que componentes trabalhem em conjunto para a formação de aplicativos distribuídos complexos, mascarando também toda complexidade para seus desenvolvedores, permitindo que a arquitetura seja baseada em múltiplas camadas de componentes.

Por este motivo, o DCOM tem uma enorme importância no desenvolvimento do trabalho, tornando possível a funcionalidade distribuída, o desenvolvimento mais claro, rápido e simplificado, agregando ainda recursos para implementação de componentes com total transparência para o usuário, que é um objetivo bastante marcante para o sistema proposto.

2.6 TRABALHOS CORRELATOS

A tese "Índices de carga e desempenho em ambientes paralelos/distribuídos – modelagem e métricas", defendida por Branco (2004), trata o problema de obtenção de um índice de carga ou de desempenho adequado para utilização no escalonamento de processos em sistemas computacionais heterogêneos paralelos/distribuídos. É feita ainda uma análise

crítica que serve de base para comparação de métricas existentes e uma nova métrica é proposta e resultados de sua aplicação são apresentados e discutidos.

Schlemer e Geyer (1998) escreveram o artigo “Escalonamento em Sistemas Distribuídos (Uma Introdução)” que fala sobre vários conceitos relacionados ao escalonamento de processos, desde como são obtidas as informações de carga de cada nodo pertencente a um determinado grupo de máquinas que se deseja escalonar, até as políticas utilizadas para o desenvolvimento de um algoritmo desse tipo, inclusive as políticas que serão utilizadas neste trabalho. É feito ainda um estudo de caso envolvendo dois sistemas operacionais distribuídos distintos, onde são apontados os pontos fracos e fortes de cada um, demonstrando os benefícios da utilização do escalonamento de processos e os problemas que um algoritmo mal desenvolvido pode causar.

Pretendendo demonstrar a importância da utilização de escalonamento de processos quando da utilização de sistemas distribuídos, Bodart et al (2004) apresentou no Workshop de Sistemas Computacionais de Alto Desempenho, o seu estudo “Avaliação de Desempenho dos Componentes de Políticas de Escalonamento de Processos em Ambientes Distribuídos”, que consistiu basicamente na avaliação de três políticas que adotam diferentes métricas de escalonamento, demonstrando os problemas de cada uma delas, calculando-se o resultado baseado no tempo de execução de cada um dos algoritmos, apontando qual teve o resultado mais adequado ao final do trabalho.

Todos os trabalhos citados tratam do mesmo tema proposto: escalonamento de processos; e tem como objetivo comum encontrar maneiras de diminuir o tempo de resposta das aplicações e maximizar o uso dos equipamentos servidores envolvidos, descrevendo inclusive as políticas que serão agregadas para o desenvolvimento deste trabalho.

3 DESENVOLVIMENTO DO TRABALHO

O sistema desenvolvido está baseado em um modelo que se concentra na utilização dos dois tipos de políticas expostos anteriormente na seção 2.3, mas que identifique a carga de processamento de cada membro do grupo a escalonar antes de encaminhar uma nova requisição aos servidores. Para tanto, a implementação de um componente intermediário que mantém diversas informações (desde o cadastro dos servidores de aplicações até o nível de carga atual suportado por cada um) que são utilizadas para tratar da questão do escalonamento dos processos é o ponto de concentração de esforços básico do trabalho.

A proposta consiste em manter registros sobre a atividade das máquinas responsáveis por atender as requisições, centralizados no componente de escalonamento, que quando recebe qualquer tipo de requisição das aplicações clientes, consulta as informações que possui para determinar o servidor ideal para atendimento da requisição. Depois disso, atualiza sua própria base, incluindo o novo processo que é recebido pelo servidor selecionado, para montar novamente uma lista contendo os dados atualizados sobre a carga dos servidores. Nessa etapa, as informações necessárias sobre a localização da máquina que deve atender a requisição efetuada anteriormente são enviadas para a aplicação cliente que necessita executar a função. Em posse desse conjunto de dados, o cliente, já devidamente direcionado, encaminha sua solicitação diretamente ao servidor que hipoteticamente encontra-se em melhores condições de atendê-la.

Para entender a junção das políticas descritas, se faz necessário então visualizar o processo mais claramente e direcionado para tal. Como o componente de escalonamento mantém atualizados os dados sobre os servidores, conhecendo o nível de carga de processamento de cada um (política de informação) e, a partir desses dados, sorteia o servidor que possui maior disponibilidade (política de localização), utilizando o resultado para montar um caminho que deve ser seguido pelo cliente solicitante, temos no contexto a utilização das duas políticas para enfatizar um modelo de escalonamento.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O Quadro 1 apresenta os requisitos funcionais levantados para a elaboração do sistema de escalonamento, bem como a sua rastreabilidade, ou seja, a devida vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deverá possuir uma aplicação do tipo cliente destinada a permitir ao usuário cadastrar as máquinas que são os servidores de aplicações e suas configurações.	UC01.01, UC02.01
RF02: O sistema deverá ter a possibilidade de receber requisições de aplicações clientes.	UC03.01
RF03: O sistema deverá manter um registro dos servidores de aplicação pertencentes ao grupo que participará do escalonamento para identificar quais deles podem atender à requisição efetuada pelo cliente	UC02.01
RF04: O sistema deverá identificar o servidor que se encontra com a menor carga de processamento no momento de uma requisição, identificando a quantidade de processos em execução, atribuindo tarefas às máquinas ociosas e retirando processos da fila quando estes terminam.	UC02.02, UC02.03
RF05: O sistema deverá enviar os dados do servidor com as melhores condições de atender a requisição para que o cliente efetue a chamada da função diretamente a este servidor.	UC02.02, UC04.01
RF06: O sistema deverá manter informações sobre as tarefas envolvidas na execução do escalonador (*log*).	UC02.04
RF07: O sistema deverá ser capaz de gerar um relatório de visualização do *log* gravado pelo escalonador.	UC03.02

Quadro 1: Requisitos funcionais

O Quadro 2 lista os requisitos não funcionais pertencentes ao sistema.

Requisitos Não Funcionais
RNF01: O sistema deverá ser implementado utilizando a filosofia de cliente/servidor.
RNF02: O sistema deverá utilizar a tecnologia *COM/DCOM* para a comunicação entre os objetos distribuídos.

Quadro 2: Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Para elaboração dos diagramas pertencentes ao processo de análise do sistema foi utilizada a ferramenta Enterprise Architect 6.0 que suporta UML 2.0 e é distribuído pela empresa Sparx Systems.

Com o apoio desta ferramenta serão apresentados a seguir os diagramas de análise de processos, pacotes, casos de uso, classes e o diagrama de implantação, dando um esboço abrangente sobre a finalidade do sistema de escalonamento de requisições.

3.2.1 Diagrama de análise de processos

O diagrama de análise a seguir contém os passos iniciais relacionados ao problema de escalonamento de requisições. Nele é demonstrado o fluxo da atividade em primeiro nível, ou seja, em sua forma mais simples, sem detalhamentos. Veja Figura 2.

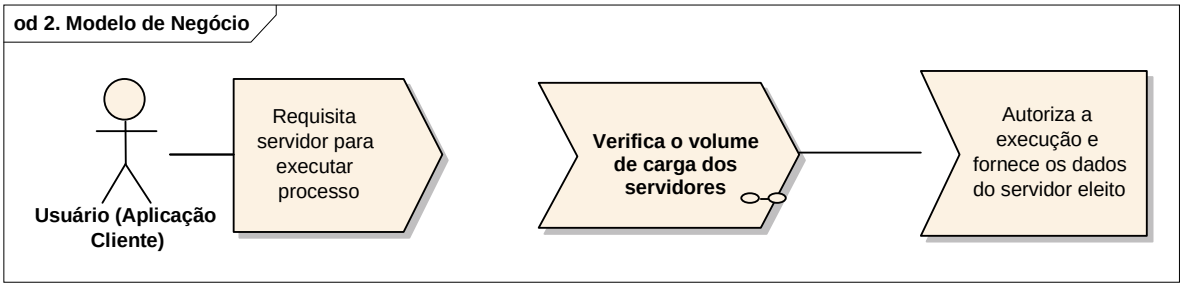

Figura 2 – Diagrama de análise de processos nível I

A seguir é inserido um pequeno detalhamento, de segundo nível, ao diagrama da atividade principal do escalonador. Como pode ser visualizada no diagrama da Figura 3, a atividade de verificação de carga recebe um pequeno refinamento para uma visão um pouco mais abrangente.

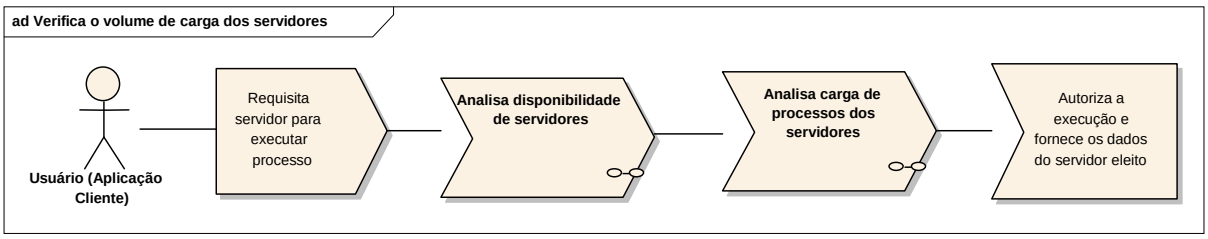

Figura 3 – Diagrama de análise de processos nível II

De forma a demonstrar os processos em um nível de refinamento maior, torna-se necessário demonstrar numa visão mais micro as atividades de análise de disponibilidade e de carga de processos nos servidores. Neste nível é que ocorre a real designação de um determinado membro para atendimento da requisição efetuada pela aplicação cliente. A Figura 4 demonstra o fluxo do processo de análise de disponibilidade dos servidores.

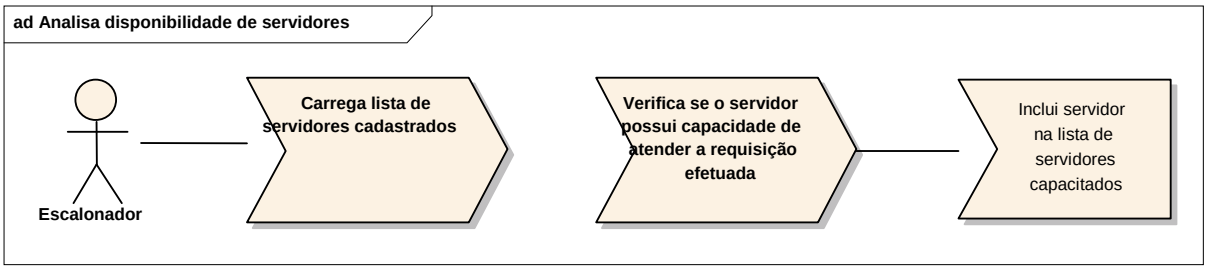

Figura 4 – Diagrama de análise de processos nível III – Detalhamento da função de análise de disponibilidade dos servidores

Na figura 5 encontra-se descrita a função de análise de carga de processos, atividade esta que verifica os processos executando em cada máquina membro do grupo de escalonamento.

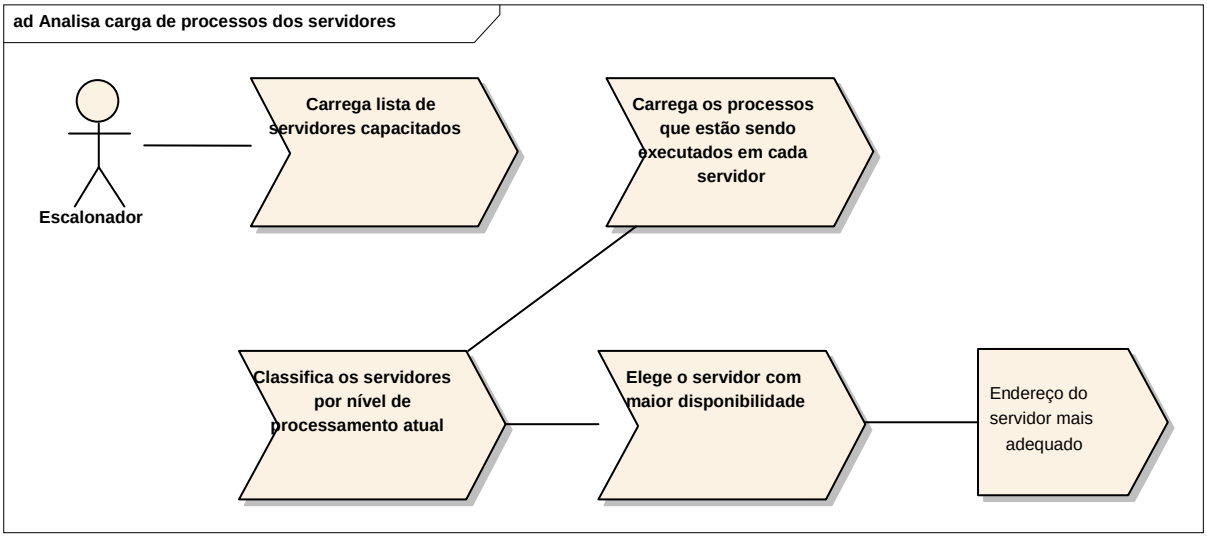

Figura 5 – Diagrama de análise de processos nível III – Detalhamento da função de análise de carga de processos dos servidores

3.2.2 Diagramas de casos de uso

Esta seção aborda os casos de uso que fazem parte do sistema, sendo que o detalhamento de cada um deles encontram-se no Apêndice A.

Para uma melhor apresentação, os casos de uso foram divididos em pacotes de acordo com os quatro componentes que fazem parte do sistema, demonstrando assim o funcionamento separado de cada elemento. Sendo assim, foi criado também um digrama de pacotes que será apresentado a seguir.

3.2.2.1 Diagrama de pacotes

Como o sistema está dividido em quatro componentes distintos, o diagrama de pacotes a seguir (Figura 6) demonstra primeiramente quais são estes componentes, para que posteriormente sejam apresentados em detalhes os diagramas de casos de uso de cada um deles.

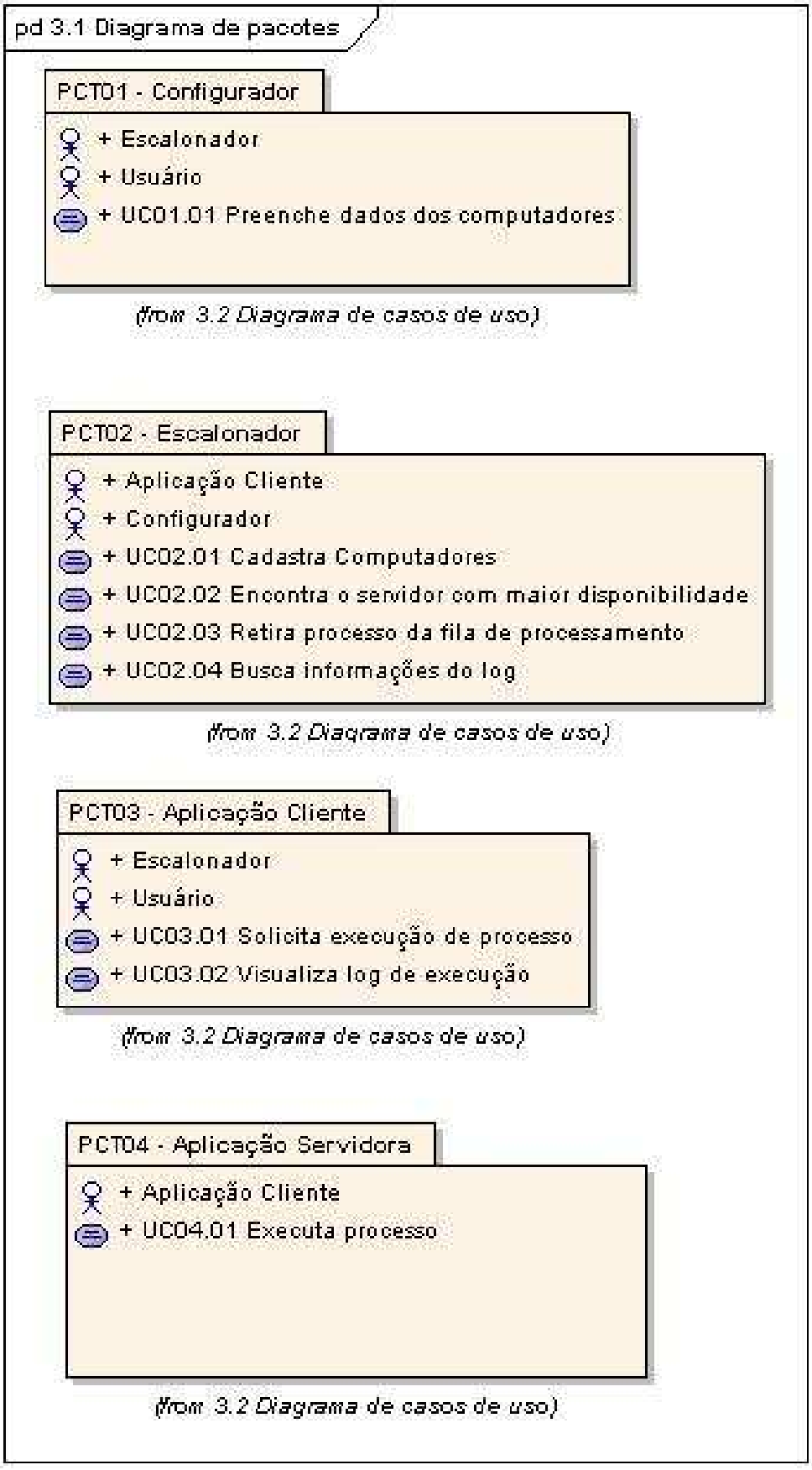

Figura 6 – Diagrama de pacotes

3.2.2.2 PCT01 - Configurador do Sistema

O componente configurador do sistema é responsável por receber os dados informados pelo usuário sobre a configuração das máquinas que participam do processo de escalonamento de requisições, e o caso de uso correspondente está demonstrado na Figura 7.

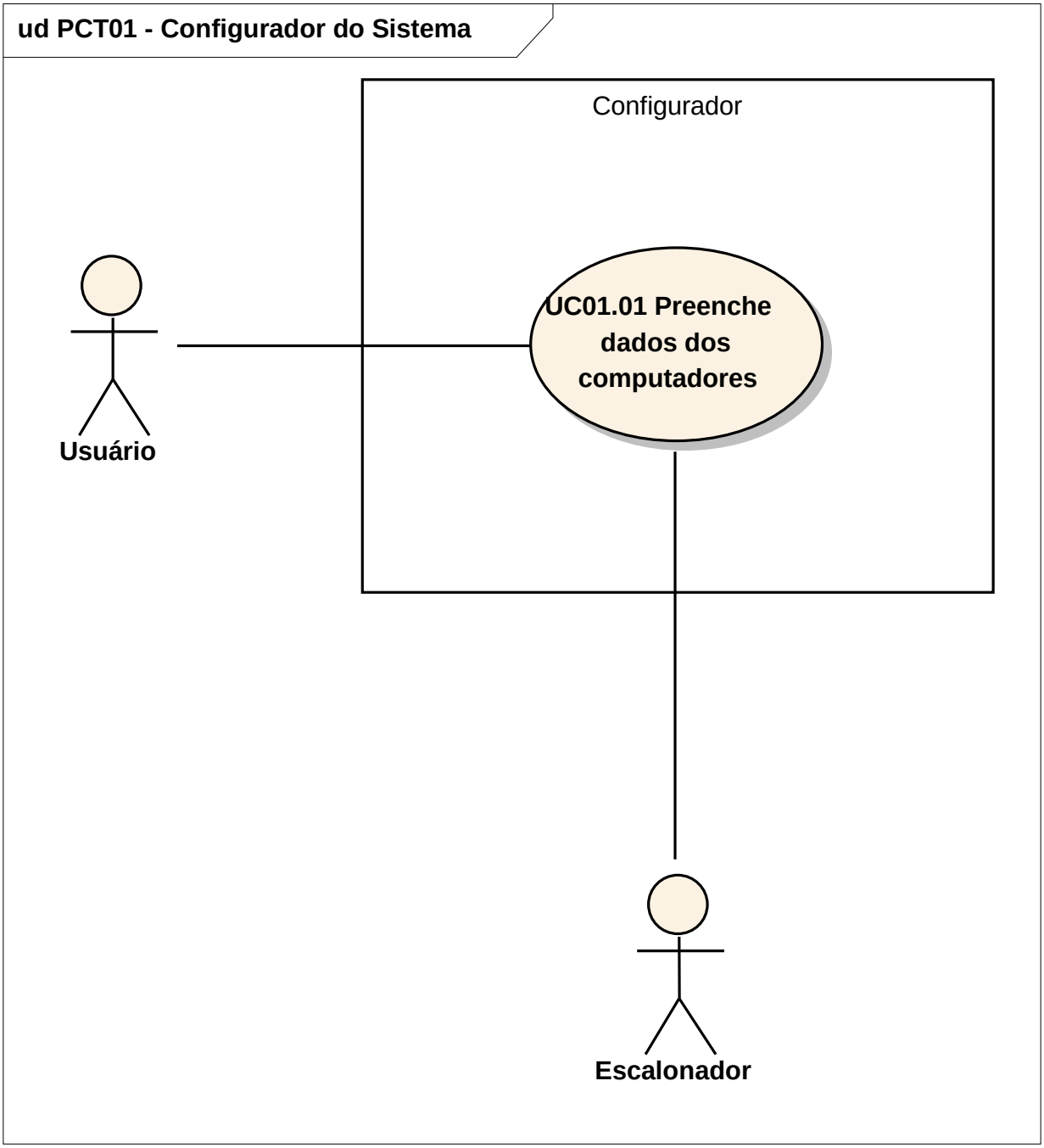

Figura 7 – Diagrama de casos de uso do configurador do sistema

3.2.2.3 PCT02 – Escalonador

O escalonador possui quatro casos de uso relacionados, que são operações requisitadas pelo configurador e pela aplicação cliente, como pode ser observado na Figura 8.

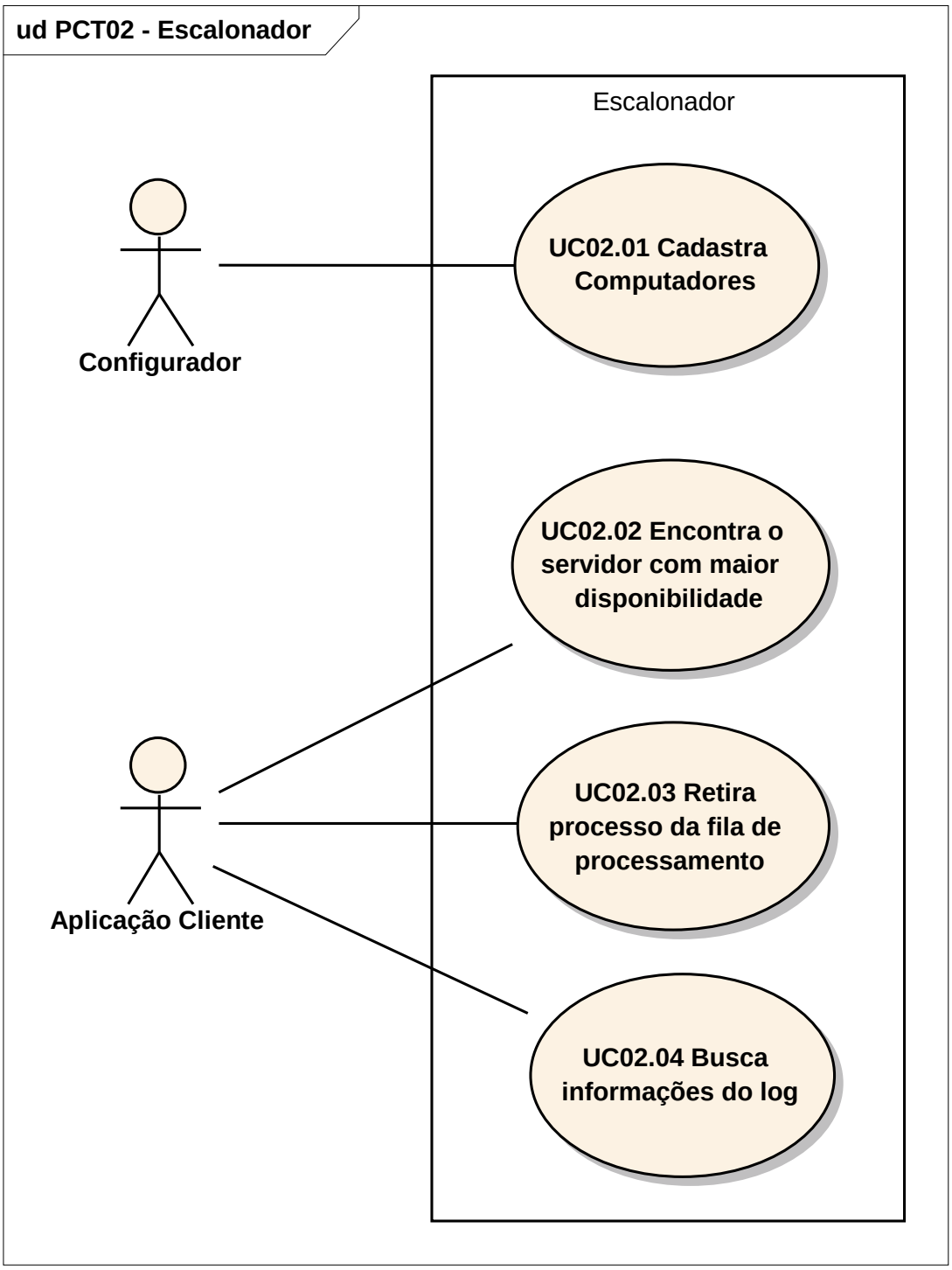

Figura 8 – Diagrama de casos de uso do escalonador

3.2.2.4 PCT03 - Aplicação Cliente

Os casos de uso relacionados com a aplicação cliente podem ser visualizados através da figura 9. Ambos são solicitações diretas do usuário e estão ligados ao escalonador, que é o responsável por atender essas requisições.

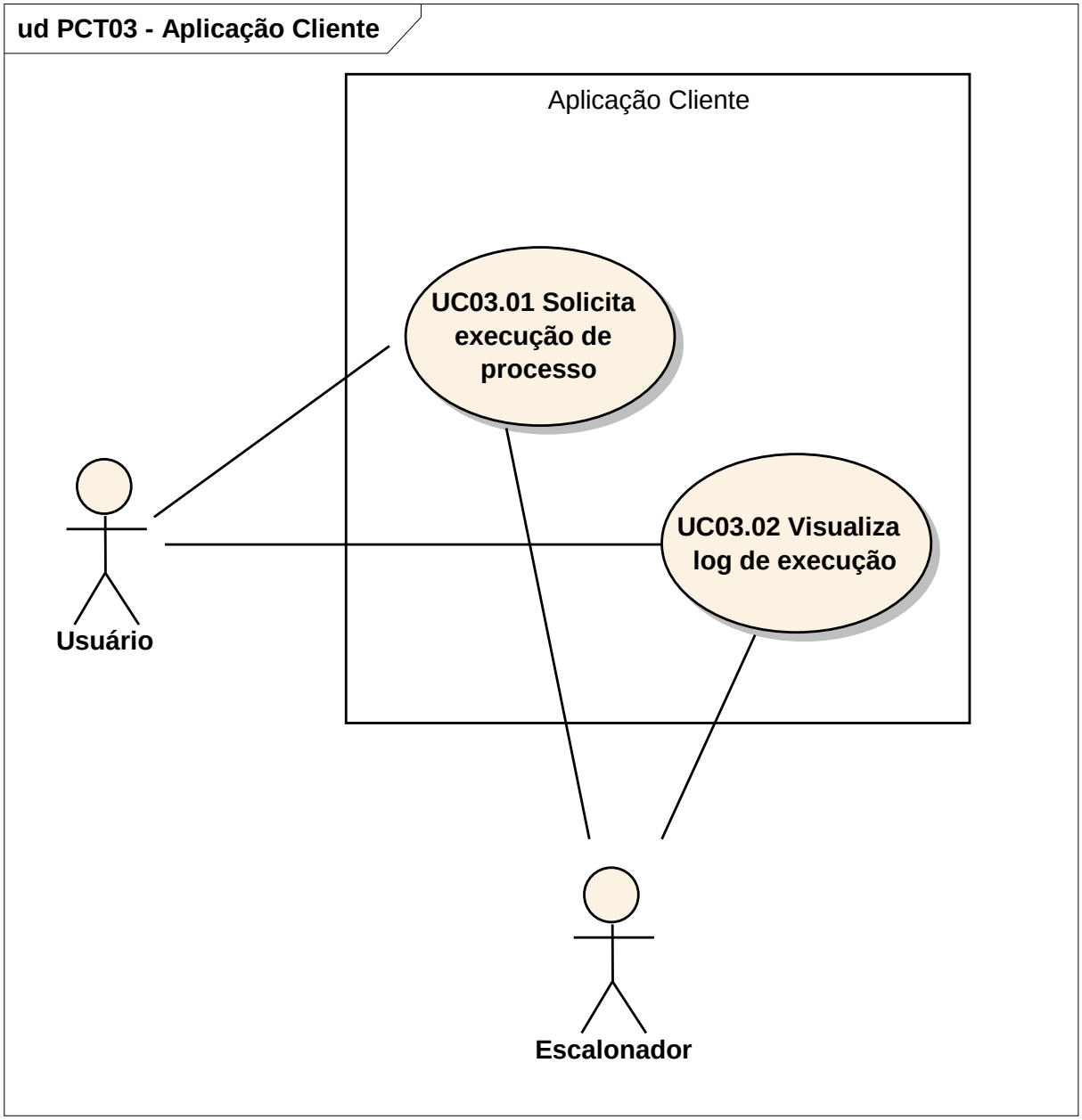

Figura 9 – Diagrama de casos de uso da aplicação cliente

3.2.2.5 PCT04 - Aplicação Servidora

A figura 10 demonstra o caso de uso relacionado com a aplicação servidora, onde a aplicação cliente solicita a execução de um determinado processo.

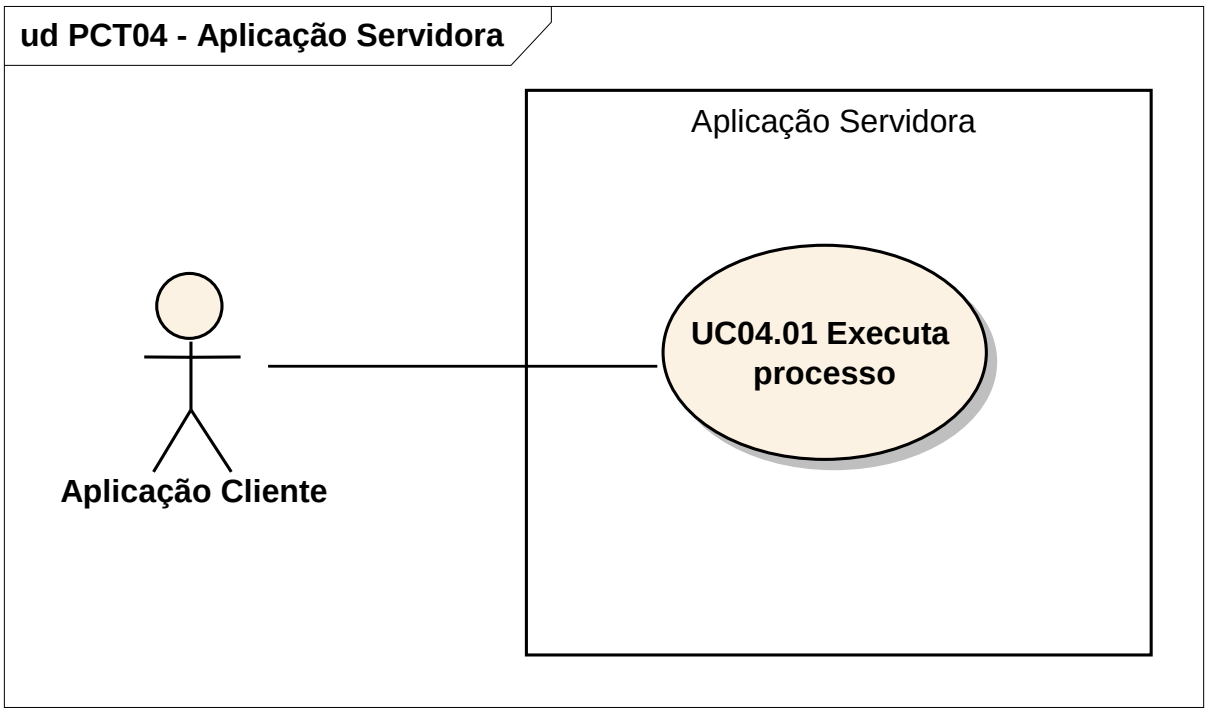

Figura 10 – Diagrama de casos de uso da aplicação servidora

3.2.3 Diagrama de classes

Nos diagramas a seguir estão descritos os métodos e propriedades relacionadas às classes contidas nos aplicativos pertencentes ao ambiente de escalonamento de requisições. Seguindo o padrão escolhido para demonstração dos casos de uso, os diagramas de classes também foram divididos em pacotes para uma melhor visualização e detalhamento individual de suas funcionalidades.

Cada seção descreve um componente que faz parte do projeto, e algumas explicações sobre o funcionamento de determinadas classes foi inserido como nota dentro do próprio diagrama, para evidenciar a utilidade das mesmas.

3.2.3.1 PCT01 - Configurador do Sistema

O configurador do sistema utiliza um ambiente cliente/servidor, onde a comunicação com o escalonador (servidor que se encarrega da manipulação dos dados fornecidos à este componente pelo usuário) é feita utilizando-se da tecnologia DCOM. Na figura 11, abaixo, é possível visualizar as classes que compõe este componente.

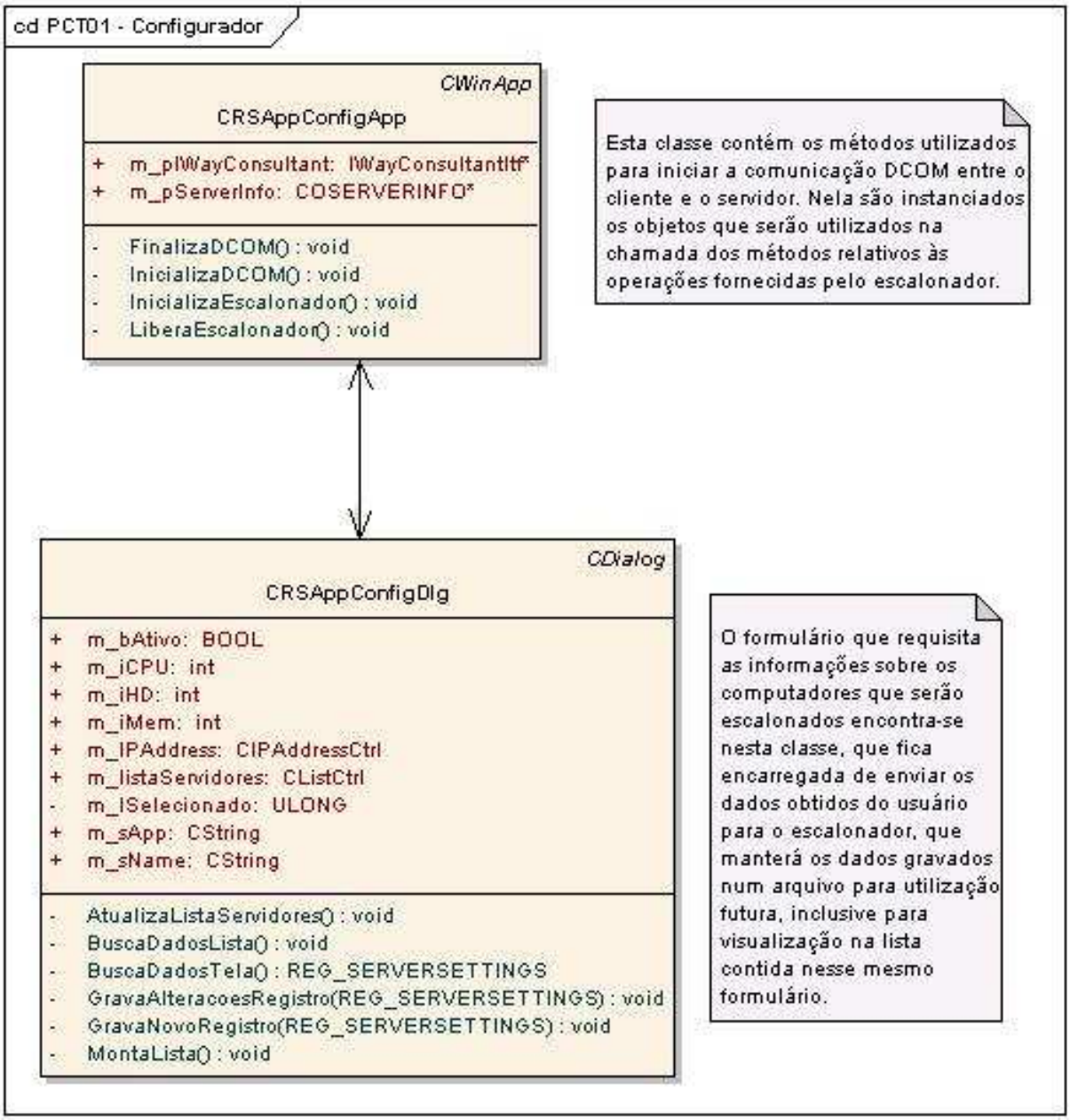

Figura 11 – Diagrama de classes do configurador do sistema

3.2.3.2 PCT02 – Escalonador

Sendo o componente principal do sistema, contém as rotinas de escalonamento das requisições e obtenção de informações das máquinas do grupo. É um componente do tipo servidor e utiliza, como nos demais componentes, a tecnologia DCOM para comunicação. Na figura 12 encontra-se o diagrama das classes pertencentes a este componente, bem como das estruturas utilizadas para armazenamento dos dados necessários ao funcionamento do escalonador.

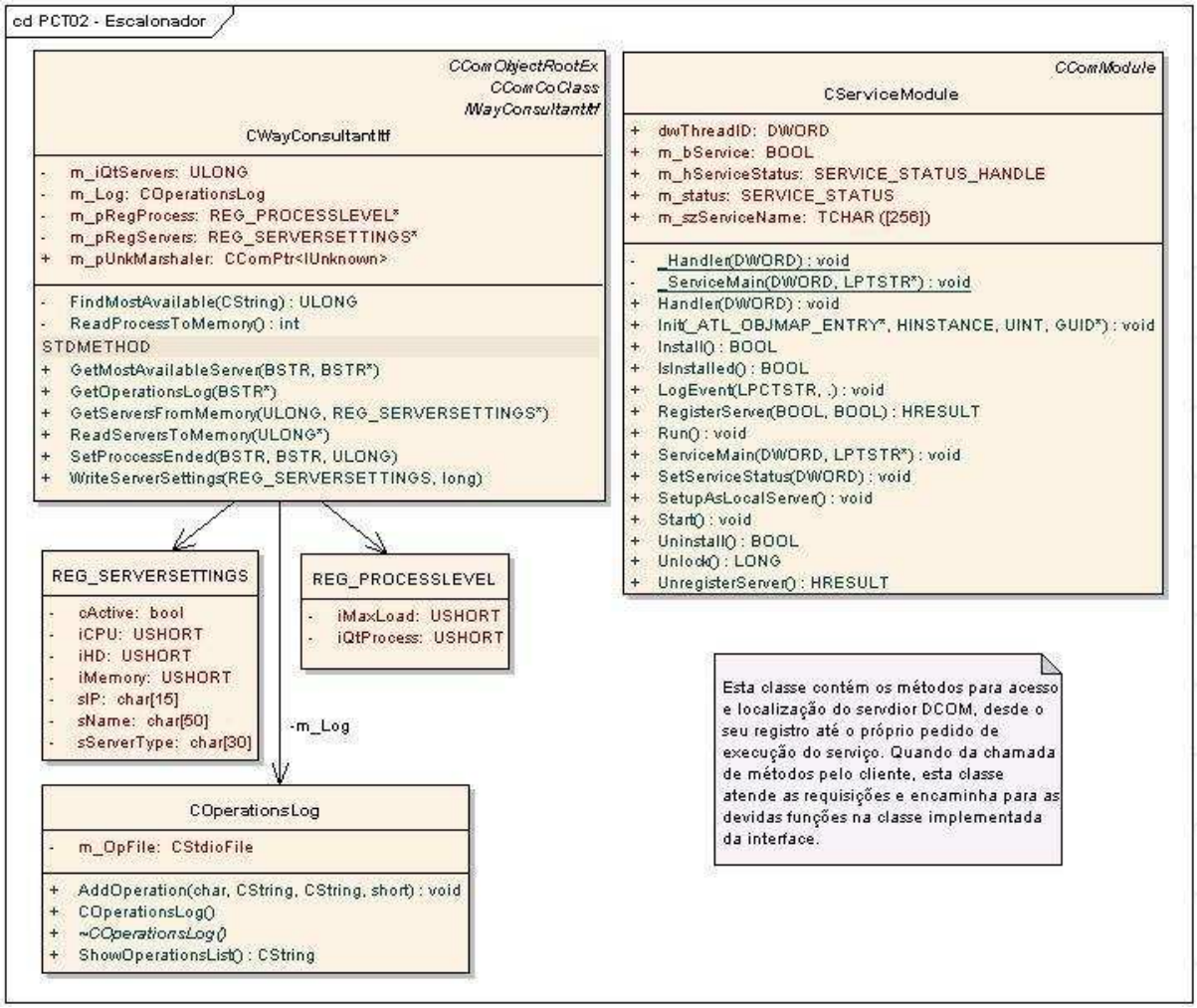

Figura 12 – Diagrama de classes do escalonador

3.2.3.3 PCT03 – Aplicação Cliente

A figura 13 demonstra as classes da aplicação cliente e dá uma breve explanação sobre o funcionamento e utilidade de cada uma delas.

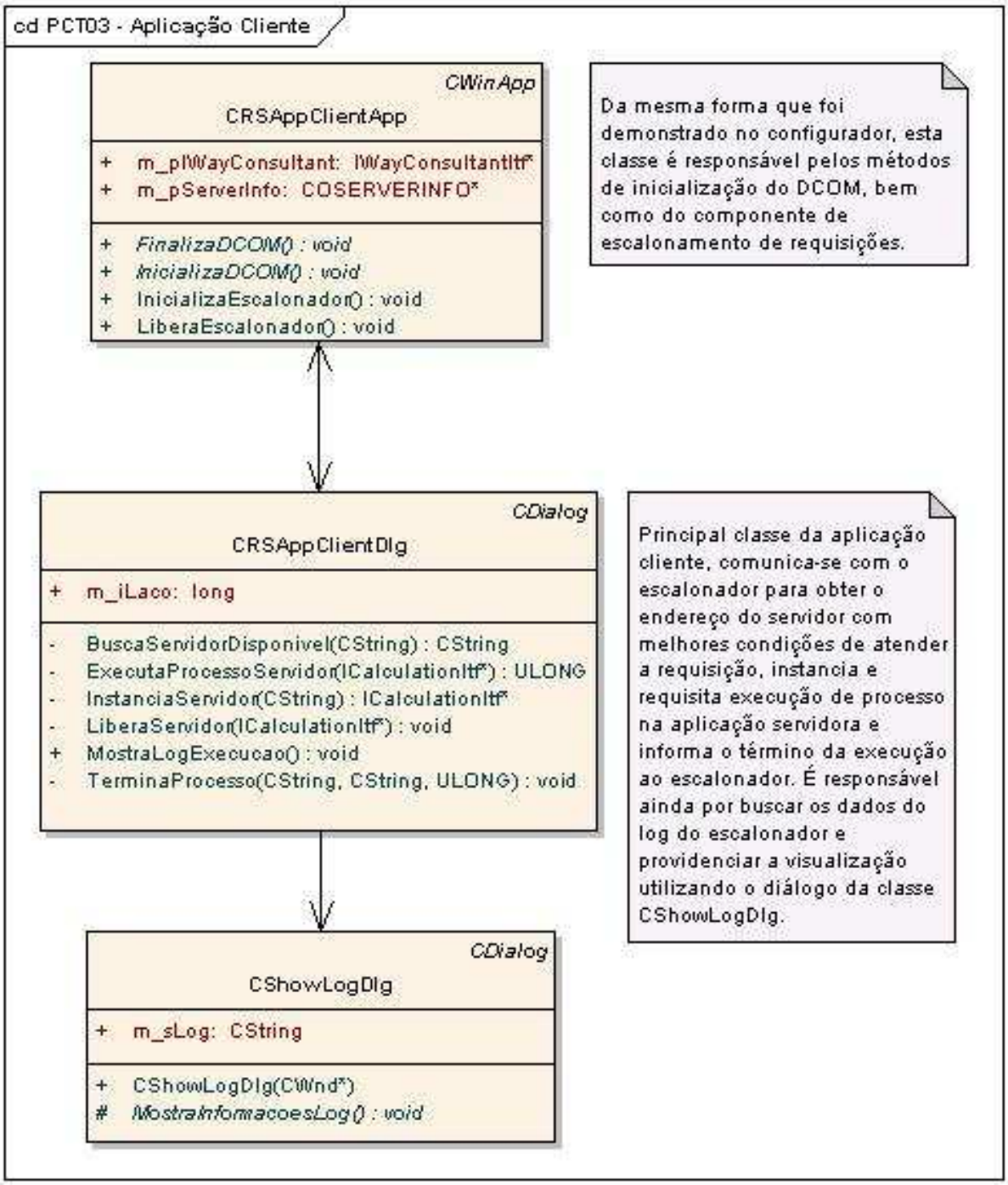

Figura 13 – Diagrama de classes da aplicação cliente

3.2.3.4 PCT04 – Aplicação Servidora

Criada com a intenção de fornecer um método para teste do ambiente de escalonamento, a aplicação servidora contém apenas esta informação e a classe que implementa a comunicação entre os componentes DCOM, conforme diagrama da figura 14.

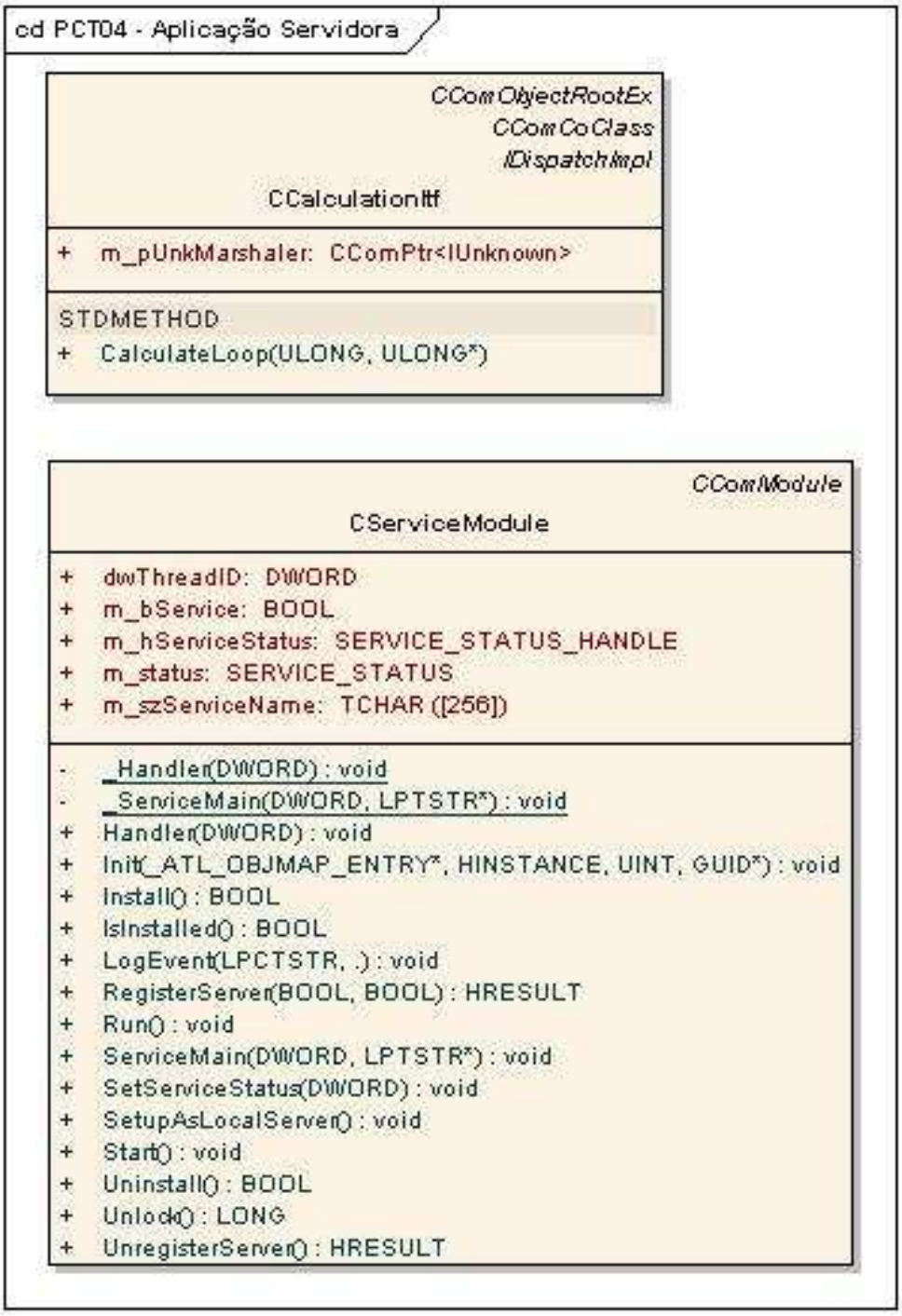

Figura 14 – Diagrama de classes da aplicação servidora

3.2.4 Diagrama de implantação

Este diagrama demonstra a distribuição dos componentes no sistema, sendo que podem existir várias máquinas executando a aplicação servidora e outras tantas executando a aplicação cliente, porém apenas uma máquina poderá executar o escalonador e, apesar de existir a possibilidade, é aconselhável também que apenas uma execute o configurador do sistema. A figura 15 demonstra uma proposta de sua distribuição máxima.

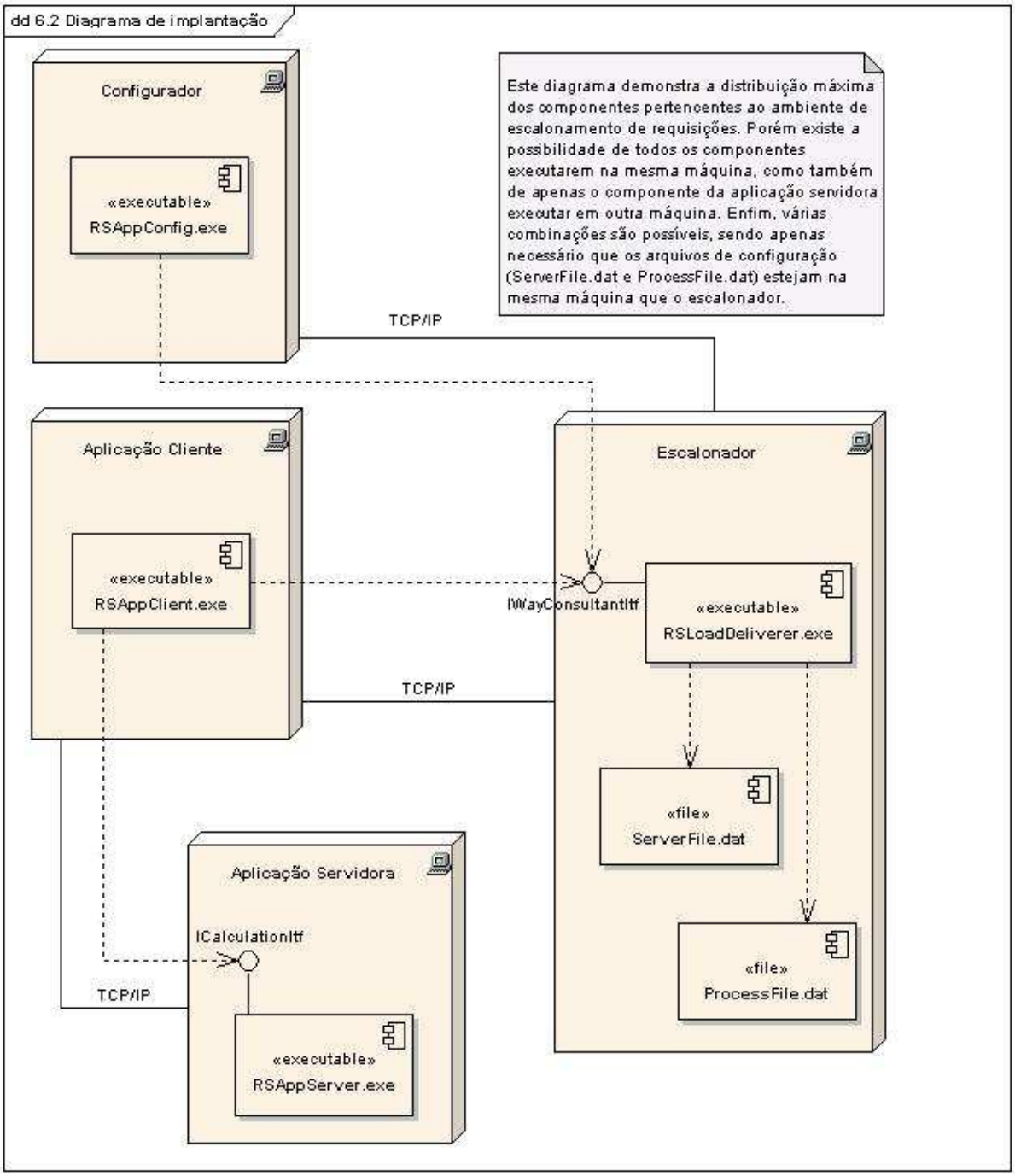

Figura 15 – Diagrama de implantação

3.3 IMPLEMENTAÇÃO

Na implementação deste trabalho foram desenvolvidos quatro aplicativos distintos:

a) cliente de configuração dos servidores: este aplicativo tem como objetivo possibilitar o usuário administrador do sistema cadastrar as informações básicas necessárias para o funcionamento do escalonador de requisições. Com o auxílio desta ferramenta, o administrador do sistema informa os servidores disponíveis para cada aplicação em que o escalonamento das requisições será efetuado. Além de uma descrição do servidor e do endereço IP da máquina, são solicitadas as informações de tipo de aplicação suportada e as informações básicas de configuração de hardware da máquina, tais como velocidade do processador, quantidade de memória RAM e capacidade total do disco rígido. Todas essas informações são fundamentais para o perfeito funcionamento do escalonador de requisições.

b) escalonador de requisições: o escalonador de requisições, propriamente dito, é o aplicativo que controla as informações de carga de processamento dos servidores do grupo. Tem como objetivo principal definir quais os nodos mais adequados para atendimento das requisições das aplicações clientes. Para tanto, obtém sempre dados atualizados sobre a carga de processamento de cada servidor e utilizando suas informações de configuração de hardware, efetua cálculos que definem qual o servidor mais adequado para atender cada requisição. Desta forma, todas as requisições são registradas por este componente, desde quando uma tarefa é dirigida para um determinado servidor até quando esta tarefa termina, liberando carga de processamento.

c) cliente de aplicação: este é o componente do sistema que obtém os benefícios oferecidos pelo escalonador de requisições. Nele estão todas as chamadas ao sistema de escalonamento, solicitando a execução de funções e obtendo as informações de acesso ao servidor selecionado para atender a requisição.

d) servidor de aplicação: este é o programa que processa as informações para atendimento das tarefas solicitadas pela aplicação cliente. Contém algoritmos utilizados para resolver determinados tipos de problemas que estão divididos em funções específicas para cada caso e que são chamadas pelos clientes com o

intuito de que os servidores se encarreguem de processar os dados e devolver apenas os resultados esperados.

3.3.1 Técnicas e ferramentas utilizadas

Para implementação do sistema foram utilizadas as ferramentas de desenvolvimento do pacote Visual Studio 6.0, mais especificamente o ambiente Visual C++ da Microsoft. Todas as telas foram criadas a partir desta ferramenta e o código fonte dos métodos e funções que compreendem cada componente foi escrito utilizando esta linguagem.

Para a comunicação entre os componentes operando em ambiente de processamento distribuído, foi utilizada a tecnologia COM/DCOM, devidamente enunciada e descrita em seção específica da fundamentação teórica.

Os seguintes requisitos foram analisados na escolha deste conjunto de ferramentas, sendo de grande importância para possibilitar o desenvolvimento do trabalho:

a) total suporte à tecnologia de orientação à objetos (OO);
b) perfeita integração entre a ferramenta de desenvolvimento e a tecnologia de comunicação de objetos;
c) flexibilidade e agilidade dos componentes gerados utilizando a linguagem;
d) fácil acesso a conteúdo de pesquisa relacionado.

Normalmente os sistemas desenvolvidos utilizando a tecnologia DCOM para comunicação, fazem uso de uma facilidade oferecida para localização dos componentes, baseada em uma chave gravada no *registry* do sistema operacional (Windows). Desta forma, todas as aplicações clientes que fazem uso dos métodos implementados na aplicação servidora, executam o componente servidor no local indicado por esta chave.

Para a implementação do escalonador de processos foi utilizada uma função que permite instanciar um objeto através do endereço IP do servidor, desde que o mesmo esteja devidamente registrado na máquina à qual o endereço se refere. Desta forma é possível que as aplicações servidoras executem em diversas máquinas diferentes, bastando que a aplicação cliente informe o endereço da máquina na qual deseja que o método seja executado, possibilitando assim a distribuição das requisições entre um grupo de computadores que esteja cadastrado para atendimento delas. Abaixo, na Figura 16, é possível visualizar o código implementado no sistema.

```
ICalculationItf* CRSAppClientDlg::InstanciaServidor(CString sCaminhoServidor)
{
    // Instancia o servidor eleito pelo escalonador
    CString sRetorno = "";
    COSERVERINFO svrInfo = {0};

    TRACE(sCaminhoServidor);
    TRACE("\n");

    // Monta o endereço do servidor selecionado na estrutura para chamada da função
    WCHAR wsServer[MAX_PATH];
    MultiByteToWideChar(CP_ACP, MB_PRECOMPOSED, sCaminhoServidor, -1, wsServer, MAX_PATH);
    svrInfo.pwszName = wsServer;

    // Função utilizada para instanciar a interface do servidor de cálculo
    MULTI_QI qi = {&IID_ICalculationItf, NULL, 0};
    HRESULT hr = CoCreateInstanceEx(CLSID_CalculationItf, NULL, CLSCTX_ALL, &svrInfo, 1, &qi);
    if (FAILED(hr))
    {
        sRetorno.Format("Problemas ao instanciar a interface de cálculo. Código do erro: %d", hr);
        AfxMessageBox(sRetorno);
        exit(1);
    }

    return (ICalculationItf*)qi.pItf;
}
```

Figura 16 – Código fonte da função InstanciaServidor

Desta forma foi resolvido o problema de instanciar um servidor dinamicamente, sem precisar saber o local do mesmo antes da execução.

Porém, o objetivo principal do sistema é o de balancear a carga das tarefas entre os servidores pertencentes ao grupo que se queira escalonar. Para isso é necessário encontrar o servidor que esteja com o menor nível de carga para atender à requisição. Nesse ponto é preciso conhecer o escalonador de requisições, um componente desenvolvido para este fim, que contém as informações de todas as máquinas e permanece em constante atualização no que diz respeito aos processos relacionados ao sistema que cada máquina está atendendo.

Para que isto seja possível, primeiramente o sistema precisa ter acesso às informações de configuração de cada estação, o que é feito através de um arquivo que é mantido pelo usuário e gravado por intermédio do configurador. A figura 17 contém o código utilizado para gravar e alterar o arquivo mencionado.

```
STDMETHODIMP CWayConsultantItf::WriteServerSettings(REG_SERVERSETTINGS regServer, long lPosition)
{
    // Calcula a carga máxima de processos suportados pela máquina, baseado nas informações de
    // velocidade do processador e memória instalada
    REG_PROCESSLEVEL regProcess = {0};

    float fCPURelation = regServer.iProcessor;
    float fMemoryRelation = regServer.iMemory;

    fCPURelation /= 1000;
    fMemoryRelation /= 1024;

    fCPURelation += 1;
    fMemoryRelation += 1;

    regProcess.iMaxLoad = regServer.iFactor + (fCPURelation * fMemoryRelation);
    ////////////////////////////////////////////////////////////////////////////////////////////

    CFile fileSvr, filePrc;

    if (lPosition == -1)
    {
        // Grava um novo registro contendo as informações do servidor cadastrado pelo configurador
        fileSvr.Open("C:\\ServerFile.dat", CFile::modeCreate | CFile::modeNoTruncate | CFile::modeWrite);
        fileSvr.SeekToEnd();
        fileSvr.Write(&regServer, sizeof(REG_SERVERSETTINGS));

        filePrc.Open("C:\\ProcessFile.dat", CFile::modeCreate | CFile::modeNoTruncate | CFile::modeWrite);
        filePrc.SeekToEnd();
        filePrc.Write(&regProcess, sizeof(REG_PROCESSLEVEL));
    }
    else
    {
        fileSvr.Open("C:\\ServerFile.dat", CFile::modeWrite);

        // Preenche registro alterado com as informações enviadas pelo cliente de configuração
        strcpy((char*)m_pRegServers[lPosition-1].sName, (char*)regServer.sName);
        strcpy((char*)m_pRegServers[lPosition-1].sIP, (char*)regServer.sIP);
        strcpy((char*)m_pRegServers[lPosition-1].sServerType, (char*)regServer.sServerType);
        m_pRegServers[lPosition-1].cActive = regServer.cActive;
        m_pRegServers[lPosition-1].iProcessor = regServer.iProcessor;
        m_pRegServers[lPosition-1].iMemory = regServer.iMemory;
        m_pRegServers[lPosition-1].iHD = regServer.iHD;
        m_pRegServers[lPosition-1].iFactor = regServer.iFactor;

        fileSvr.SetLength(0);
        fileSvr.Write(m_pRegServers, (m_iQtServers * sizeof(REG_SERVERSETTINGS)));

        filePrc.Open("C:\\ProcessFile.dat", CFile::modeWrite);

        m_pRegProcess[lPosition-1].iMaxLoad = regProcess.iMaxLoad;

        filePrc.SetLength(0);
        filePrc.Write(m_pRegProcess, (m_iQtServers * sizeof(REG_PROCESSLEVEL)));
    }

    fileSvr.Close();
    filePrc.Close();

    return S_OK;
}
```

Figura 17 – Código fonte do método WriteServerSettings

Ainda no contexto do método de cadastramento de máquinas para escalonamento, é possível visualizar a rotina que define a quantidade de processos relacionados com a aplicação que cada uma das máquinas pode executar. Esta quantidade é calculada utilizando-se um fator informado pelo usuário (que seria o número inicial de processos que cada máquina deve comportar), em conjunto com as informações de configuração das máquinas, que acrescerá mais uma quantidade de processos dependendo do hardware que equipa cada computador. Quanto maior a capacidade de processamento e de armazenamento em memória, maior o número de processos que cada máquina pode atender.

Desta forma o algoritmo pode ser ajustado pelo usuário, permitindo que os cálculos efetuados pelo escalonador sejam adequados à necessidade de cada grupo de máquinas onde o ambiente de escalonamento seja implantado.

Na prática, o cálculo é efetuado levando em consideração a freqüência do *clock* da CPU, em MHz (FCLK) e a capacidade de memória RAM, especificada em *MBytes* (TRAM), ajustado por intermédio do número mínimo de processos, informado pelo usuário (NPROCESSOS), da seguinte maneira:

iMaxLoad = NPROCESSOS + ((FCLK/1000 +1)*(TRAM/1024 +1)).

Os valores de FCLK e TRAM são obtidos do cadastro de servidores. Por exemplo, para um computador com processador 2.4GHz (FCLK=2400), 1GB de memória RAM (TRAM=1024), onde o fator for ajustado por NPROCESSOS = 50, tem-se:

iMaxLoad = 50 + (((2400/1000)+1) * ((1024/1024) + 1)) = 50 + (3,4 * 2) = 50 + 6,8 = 56,8 , onde existe o arredondamento por parte do sistema, resultando em *iMaxLoad* = 57.

Uma segunda configuração composta de CPU com *clock* 350MHz (FCLK= 350), 256Mb de memória RAM (TRAM = 256), NPROCESSOS = 20, tem-se:

iMaxLoad = 20 + (((350/1000)+1) * ((256/1024)+1)) = 20 + (1,35 * 1,25) = 20 + 1,6875 = 21,6875, arredondando, o valor de *iMaxLoad* será 22.

O número de processos encontrado por intermédio deste cálculo é utilizado na rotina de escalonamento das requisições como um fator determinante do percentual de ociosidade observado em cada computador, e, conseqüentemente para decidir qual máquina deve atender a requisição.

Além desse arquivo, cada requisição da aplicação cliente passa pelo escalonador, que por sua vez atualiza um arquivo de processos em execução mantido para cada máquina. O incremento de processos ocorre quando da escolha do servidor para atender a requisição e a liberação do processo é feita através da chamada da função *SetProcessEnded*, descrita a seguir na Figura 18.

```
STDMETHODIMP CWayConsultantItf::SetProccessEnded(BSTR bsServerType, BSTR bsServerAddress, ULONG lTimeExpended)
{
    // Efetua a leitura da carga de processamento atual de todos os servidores cadastrados
    ReadProcessToMemory();
    ////////////////////////////////////////////////////////////////////////////////////

    CString sServerType = bsServerType;
    CString sServerAddress = bsServerAddress;
    ULONG lTime = lTimeExpended;

    // Adiciona a operação de término de processamento no log do escalonador
    m_Log.AddOperation('T', sServerType, sServerAddress);

    for (ULONG i=0; i < m_iQtServers; i++)
    {
        // Mensagem de visualização do número de processos executando no servidor
        // Obs.: Para fins de teste! Não fica disponível ao usuário
        TRACE("  -- Servidor %s com %d processos executando.\n", m_pRegServers[i].sIP, m_pRegProcess[i].iQtProcess);

        // Exclui o processo da lista de processos executando no servidor selecionado
        if ((strcmp((char*)m_pRegServers[i].sIP, sServerAddress) == 0) &&
            (strcmp((char*)m_pRegServers[i].sServerType, sServerType) == 0))
            m_pRegProcess[i].iQtProcess--;
        ////////////////////////////////////////////////////////////////////////////////
    }

    // Grava o arquivo de controle dos servidores com as informações atualizadas dos processos
    CFile file;
    file.Open("C:\\ProcessFile.dat", CFile::modeWrite);

    file.SetLength(0);
    file.Write(m_pRegProcess, (m_iQtServers * sizeof(REG_PROCESSLEVEL)));
    file.Close();
    ////////////////////////////////////////////////////////////////////////////////////////

    return S_OK;
}
```

Figura 18 – Código fonte do método SetProcessEnded

Com a utilização destas regras, mantendo sempre as informações precisas da configuração das máquinas e dos processos em execução, resta apenas que o escalonador as utilize para encontrar o servidor com a maior disponibilidade, tarefa que pode ser visualizada através do código contido na Figura 19. Nela pode ser observada a utilização da variável *iMaxLoad*, obtida no cálculo efetuado dentro do método WriteServerSettings, mencionado anteriormente. Esta variável é a chave para que o sistema efetue o correto escalonamento das requisições.

```
ULONG CWayConsultantItf::FindMostAvailable(CString sServerType)
{
    // Efetua a leitura da carga de processamento atual de todos os servidores cadastrados
    ReadProcessToMemory();
    ////////////////////////////////////////////////////////////////////////////////////////

    ULONG iChosen = 0;
    short iLast = -32000, iDiff = 0;

    for (ULONG i=0; i<m_iQtServers; i++)
    {
        // Verifica os servidores que atendem às requisições da aplicação desejada
        // e se o mesmo está ativo
        if ((sServerType == m_pRegServers[i].sServerType) &&
            (m_pRegServers[i].cActive == 1))
        {
            // Calcula o percentual de ociosidade do servidor
            iDiff = ((m_pRegProcess[i].iMaxLoad - m_pRegProcess[i].iQtProcess) * 100) / m_pRegProcess[i].iMaxLoad;

            // Mensagem de visualização do número de processos executando no servidor
            // Obs.: Para fins de teste! Não fica disponível ao usuário
            TRACE("++ Servidor %s com %d processos executando, %d %% de ociosidade.\n",
                m_pRegServers[i].sIP, m_pRegProcess[i].iQtProcess, iDiff);

            // Inclui no log uma operação informando o percentual de ociosidade do servidor
            m_Log.AddOperation('L', m_pRegServers[i].sServerType, m_pRegServers[i].sIP, iDiff);

            if (iDiff > iLast)
            {
                iChosen = i;
                iLast = iDiff;
            }
        }
    }

    // Incrementa a quantidade de processos executando no servidor eleito
    m_pRegProcess[iChosen].iQtProcess++;

    // Grava o arquivo de controle dos servidores com as informações atualizadas dos processos
    CFile file;
    file.Open("C:\\ProcessFile.dat", CFile::modeWrite);

    file.SetLength(0);
    file.Write(m_pRegProcess, (m_iQtServers * sizeof(REG_PROCESSLEVEL)));

    file.Close();
    ////////////////////////////////////////////////////////////////////////////////////////

    return iChosen;
}
```

Figura 19 – Código fonte do método FindMostAvailable

3.3.2 Operacionalidade da implementação

Esta seção demonstra como operar o sistema e quais as funções atendidas por ele.

Para utilização do ambiente de escalonamento de requisições, alguns passos devem ser seguidos a fim de tornar a operação possível.

3.3.2.1 Cadastramento das máquinas para escalonamento

Num primeiro momento, o cadastramento dos servidores participantes do processo de escalonamento deve ser efetuado por um dos usuários do sistema. Nesta etapa existe a interação com o componente denominado configurador, evidenciado na Figura 20.

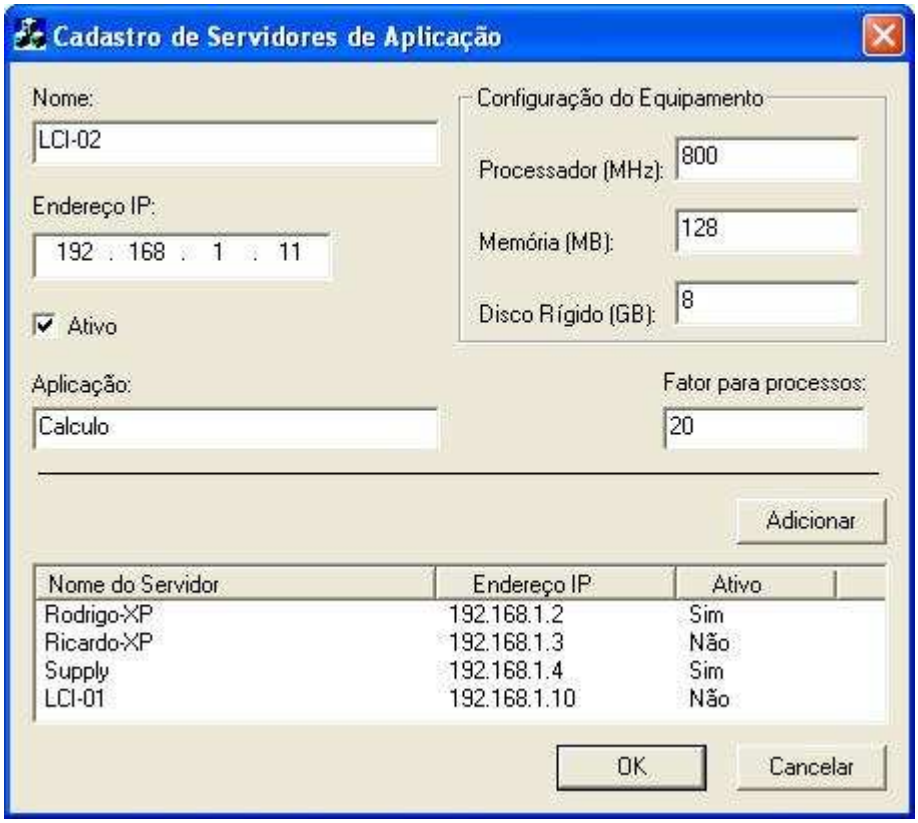

Figura 20 – Inclusão de máquinas no configurador

O componente de cadastro de servidores é responsável por fazer a interação direta com o usuário e receber as informações necessárias sobre todas as máquinas que podem fazer parte do processo de escalonamento, tais como o endereço para comunicação, a aplicação instalada nesta máquina e as informações sobre a sua configuração física (velocidade do processador, quantidade de memória RAM instalada e capacidade do disco rígido), além do fator utilizado para o cálculo da quantidade de processos envolvendo a aplicação informada a serem executados em cada máquina. Após o preenchimento dos dados, a inclusão pode ser efetuada clicando-se no botão Adicionar, operação que resulta no envio os dados ao escalonador que fica encarregado de gravar as informações num arquivo de dados para utilização futura.

Se por ventura o usuário precisar alterar algum desses dados ou desativar uma máquina cadastrada, excluindo a mesma do processo de escalonamento, é possível fazê-lo selecionando a máquina desejada na lista de exibição através de um duplo clique. As informações de configuração da máquina serão apresentadas na tela e pode sofrer as alterações desejadas, bastando que o usuário acione o botão alterar para que as mesmas sejam enviadas ao escalonador que se encarregará de regravar os dados com as correções. A tela que o usuário visualiza durante esta atividade pode ser verificada na figura 21.

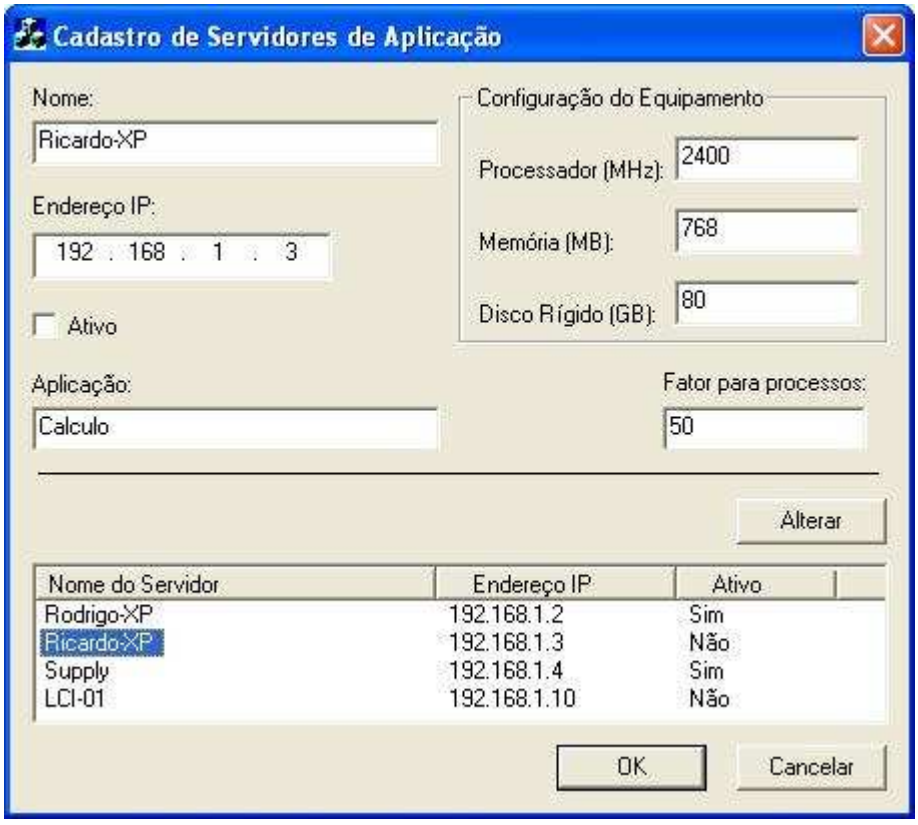

Figura 21 – Alteração de máquinas no configurador

3.3.2.2 Requisição de tarefas

Tendo todas as máquinas devidamente cadastradas, o escalonador pode finalmente entrar em ação, recebendo as requisições das aplicações cliente e escalonando-as para os computadores com maior disponibilidade de atendê-las.

O sistema foi construído de forma a possibilitar sua utilização por diferentes aplicações de controle, porém, para uma demonstração do seu funcionamento foi desenvolvido um componente simples com as funcionalidades necessárias de comunicação através de interfaces DCOM, chamado de componente de cálculo. A figura 22 contém um esboço da tela da aplicação cliente, utilizada para essa demonstração.

Figura 22 – Aplicação cliente de cálculo

Com o auxílio desta aplicação, o usuário interage com o sistema, determinando que o mesmo execute um número finito de passagens por uma rotina de cálculo simples. A quantidade de vezes que a aplicação servidora efetuará os cálculos fica definida pelo tamanho do laço informado pelo usuário.

Após a informação desta quantidade, clicando-se no botão requisita tarefa, o sistema envia uma requisição ao escalonador, que verifica quais computadores estão capacitados de processar esta tarefa e qual deles é o mais indicado ou que se encontra com maior disponibilidade no momento do pedido. De posse dessas informações, o escalonador envia o endereço IP do computador selecionado para a aplicação cliente, que deve solicitar a execução da tarefa diretamente ao servidor definido.

Quando o servidor de aplicação termina de processar a tarefa, a informação do tempo gasto na execução é enviada à aplicação cliente, que deve informar ao escalonador que a tarefa anteriormente solicitada terminou de executar e que o tempo gasto nesse processo foi àquele informado pelo servidor de aplicação. A partir destas informações, o escalonador atualiza a quantidade de processos executando no computador em questão para que mais tarde seja possível obter informações precisas sobre o seu estado.

3.3.2.3 Visualização do log de execução do escalonador

Durante todo o processo de escolha do servidor com maior disponibilidade para atendimento da requisição efetuada pela aplicação cliente, o escalonador grava informações sobre as funções envolvidas nesse processo. Para acompanhamento dos resultados, foi disponibilizado na aplicação cliente, um resumo destas operações, demonstrando passo-a-

passo as decisões tomadas pelo escalonador.

Este resumo, chamado de *log* do sistema, pode ser acessado através do botão Visualiza Log encontrado na tela da aplicação cliente, demonstrada anteriormente. O resultado obtido é o que está representado na Figura 23, abaixo.

Figura 23 – Visualizador do log de execução

No *log* do sistema são demonstradas as informações gravadas durante o processo de escolha do computador com maior disponibilidade, contendo o horário exato da execução de cada tarefa envolvida, desde a solicitação até o término da execução do processo, podendo ser visto o percentual de ociosidade de cada uma das máquinas no momento da escolha e a decisão tomada pelo escalonador a partir desses dados.

3.4 RESULTADOS E DISCUSSÃO

Para validação dos resultados obtidos pelo sistema, foram efetuadas diversas execuções em laboratório, simulando um ambiente de trabalho típico, com diversas configurações de hardware distintas. As informações de configuração de cada máquina utilizada nos testes estão contidas na Tabela 1.

Tabela 1 – Configuração das máquinas utilizadas nos testes em laboratório

Descrição / IP	**Processador (MHz)**	**Memória (Mb)**	**Disco Rígido (Gb)**
D102-040 (172.16.1.216)	1466	512	20
D102-043 (172.16.1.222)	475	196	6
D102-044 (172.16.1.202)	450	256	5
D102-049 (172.16.1.209)	475	196	5
D102-050 (172.16.1.208)	450	196	5
D102-051 (172.16.1.189)	450	196	5

Foram efetuadas quatro baterias de testes utilizando parâmetros diferentes para a execução do sistema, desde a utilização de máquinas diferentes até a mudança do fator para cálculo do número de processos para cada máquina.

No apêndice B podem ser visualizadas graficamente as informações resumidas sobre cada uma dessas baterias de testes, com detalhes por execução e por máquina.

Na primeira execução foram utilizadas todas as máquinas descritas na Tabela 1, como servidores de aplicação, sendo que o computador descrito como D102-044 e com endereço IP 172.16.1.202 foi utilizado ainda para as tarefas de escalonamento e executou várias instâncias da aplicação cliente.

Os resultados desta primeira execução podem ser observados na Tabela 2.

Tabela 2 – Tempos de execução das requisições na primeira bateria de testes

Ação	Início	Fim	Tempo Gasto
Operação de cálculo no servidor 172.16.1.202	15:08:52	15:09:51	00:59
Operação de cálculo no servidor 172.16.1.208	15:08:52	15:09:49	00:57
Operação de cálculo no servidor 172.16.1.189	15:08:53	15:09:49	00:56
Operação de cálculo no servidor 172.16.1.209	15:08:54	15:09:47	00:53
Operação de cálculo no servidor 172.16.1.216	15:08:55	15:09:17	00:22
Operação de cálculo no servidor 172.16.1.222	15:08:55	15:09:48	00:53
Operação de cálculo no servidor 172.16.1.202	15:08:57	15:11:41	02:44
Operação de cálculo no servidor 172.16.1.208	15:08:57	15:10:44	01:47
Operação de cálculo no servidor 172.16.1.189	15:08:58	15:10:45	01:47
Operação de cálculo no servidor 172.16.1.209	15:08:59	15:10:40	01:41

Nesta primeira execução é possível observar que a máquina escolhida para executar as instâncias da aplicação cliente e também o escalonador de requisições, teve seu rendimento comprometido, pelo fato de utilizar seus recursos para atender estas tarefas além da execução das operações por parte da aplicação servidora.

Na segunda execução foram utilizadas apenas as máquinas D102-040, D102-044 e D102-050 como servidores de aplicação, tendo ainda a máquina D102-043 executando o escalonador e a mesma máquina D102-043 e a D102-044 com várias instâncias do cliente de aplicação.

Nesta bateria de testes, o fator para cálculo do número de processos a executar em cada máquina foi ajustado para 50.

Com este teste fica visível que a aplicação cliente é responsável pela maior parte do consumo dos recursos do sistema, sendo que a máquina responsável por executar as várias instâncias desta aplicação, além de atender as requisições como aplicação servidora teve a *performance* bastante comprometida, chegando a gastar quatro vezes mais tempo para atender as requisições efetuadas. Este fato tem relação com os eventos gerados pela interação do usuário com a máquina, sendo que são atendidas pelo sistema operacional antes das tarefas originadas pela aplicação servidora.

Na Tabela 3 podem ser visualizados os tempos de execução das tarefas obtidos na segunda bateria de testes.

Tabela 3 – Tempos de execução das requisições na segunda bateria de testes

Ação	Início	Fim	Tempo Gasto
Operação de cálculo no servidor 172.16.1.202	15:24:02	15:27:52	03:50
Operação de cálculo no servidor 172.16.1.216	15:24:03	15:24:25	00:22
Operação de cálculo no servidor 172.16.1.202	15:24:04	15:27:54	03:50
Operação de cálculo no servidor 172.16.1.208	15:24:04	15:25:01	00:57
Operação de cálculo no servidor 172.16.1.216	15:24:05	15:24:47	00:42
Operação de cálculo no servidor 172.16.1.202	15:24:06	15:27:55	03:49
Operação de cálculo no servidor 172.16.1.208	15:24:08	15:25:57	01:49
Operação de cálculo no servidor 172.16.1.216	15:24:14	15:25:09	00:55
Operação de cálculo no servidor 172.16.1.202	15:24:15	15:27:59	03:44
Operação de cálculo no servidor 172.16.1.208	15:24:16	15:26:52	02:36
Operação de cálculo no servidor 172.16.1.216	15:24:18	15:25:30	01:12
Operação de cálculo no servidor 172.16.1.202	15:24:18	15:28:57	04:39
Operação de cálculo no servidor 172.16.1.208	15:24:19	15:27:48	03:29
Operação de cálculo no servidor 172.16.1.216	15:24:20	15:25:52	01:32
Operação de cálculo no servidor 172.16.1.202	15:24:20	15:29:54	05:34
Operação de cálculo no servidor 172.16.1.208	15:24:21	15:28:44	04:23
Operação de cálculo no servidor 172.16.1.216	15:24:22	15:26:14	01:52
Operação de cálculo no servidor 172.16.1.202	15:24:23	15:30:52	06:29
Operação de cálculo no servidor 172.16.1.208	15:24:24	15:29:40	05:16
Operação de cálculo no servidor 172.16.1.216	15:24:24	15:26:36	02:12
Operação de cálculo no servidor 172.16.1.216	15:27:42	15:28:04	00:22
Operação de cálculo no servidor 172.16.1.216	15:27:42	15:28:25	00:43
Operação de cálculo no servidor 172.16.1.216	15:27:43	15:28:47	01:04
Operação de cálculo no servidor 172.16.1.216	15:27:44	15:29:09	01:25
Operação de cálculo no servidor 172.16.1.208	15:27:52	15:30:35	02:43
Operação de cálculo no servidor 172.16.1.216.	15:29:30	15:29:52	00:22
Operação de cálculo no servidor 172.16.1.216.	15:29:31	15:30:14	00:43
Operação de cálculo no servidor 172.16.1.202.	15:29:32	15:31:49	02:17
Operação de cálculo no servidor 172.16.1.216.	15:29:33	15:30:36	01:03

A primeira linha da tabela demonstra a situação da tarefa solicitada à máquina que foi escalada para executar tanto aplicações clientes quanto aplicações servidoras, ficando visível o tempo relativamente alto no atendimento da mesma. Porém, observando a terceira e sexta linha da tabela, é possível verificar ainda que as requisições seguintes também foram atendidas praticamente no mesmo instante, fato atribuído ao término do atendimento das operações de interação com as aplicações clientes.

Esta execução foi a que exigiu mais recursos das máquinas, sendo que vinte e nove requisições foram efetuadas aos servidores, vinte destas praticamente ao mesmo tempo.

Na terceira execução a configuração de máquinas permaneceu a mesma utilizada na execução anterior, porém o fator para cálculo do número de processos recebeu um ajuste em seu valor para 20, podendo-se observar uma mudança significativa nos resultados.

A Tabela 4 contém os resultados observados nesta terceira execução efetuada.

Tabela 4 – Tempos de execução das requisições na terceira bateria de testes

Ação	Início	Fim	Tempo Gasto
Operação de cálculo no servidor 172.16.1.202.	15:42:21	15:44:58	02:37
Operação de cálculo no servidor 172.16.1.216.	15:42:25	15:42:47	00:22
Operação de cálculo no servidor 172.16.1.202.	15:42:26	15:45:32	03:06
Operação de cálculo no servidor 172.16.1.208.	15:42:31	15:43:28	00:57
Operação de cálculo no servidor 172.16.1.216.	15:42:32	15:43:09	00:37
Operação de cálculo no servidor 172.16.1.216.	15:42:33	15:43:30	00:57
Operação de cálculo no servidor 172.16.1.202.	15:42:33	15:45:34	03:01
Operação de cálculo no servidor 172.16.1.208.	15:42:33	15:44:24	01:51
Operação de cálculo no servidor 172.16.1.216.	15:42:34	15:43:52	01:18
Operação de cálculo no servidor 172.16.1.202.	15:42:34	15:46:21	03:47
Operação de cálculo no servidor 172.16.1.208.	15:42:35	15:45:19	02:44
Operação de cálculo no servidor 172.16.1.216.	15:42:35	15:44:14	01:39
Operação de cálculo no servidor 172.16.1.202.	15:42:35	15:47:19	04:44
Operação de cálculo no servidor 172.16.1.208.	15:42:36	15:46:15	03:39
Operação de cálculo no servidor 172.16.1.208.	15:42:36	15:47:11	04:35
Operação de cálculo no servidor 172.16.1.216.	15:42:36	15:44:36	02:00
Operação de cálculo no servidor 172.16.1.202.	15:42:37	15:48:17	05:40
Operação de cálculo no servidor 172.16.1.208.	15:42:37	15:48:06	05:29
Operação de cálculo no servidor 172.16.1.216.	15:42:38	15:44:58	02:20
Operação de cálculo no servidor 172.16.1.202.	15:42:38	15:49:16	06:38
Operação de cálculo no servidor 172.16.1.208.	15:42:38	15:49:02	06:24
Operação de cálculo no servidor 172.16.1.216.	15:42:39	15:45:19	02:40
Operação de cálculo no servidor 172.16.1.216.	15:47:49	15:48:11	00:22
Operação de cálculo no servidor 172.16.1.216.	15:47:51	15:48:33	00:42
Operação de cálculo no servidor 172.16.1.216.	15:47:52	15:48:55	01:03
Operação de cálculo no servidor 172.16.1.202.	15:47:57	15:50:19	02:22
Operação de cálculo no servidor 172.16.1.208.	15:48:28	15:49:58	01:30

Como pode ser observado nas linhas 5 e 6 da tabela, o ajuste efetuado no fator de cálculo resultou na escolha da máquina mais potente (àquela com endereço IP 172.16.1.216) com uma maior freqüência, situação que ocasionou uma melhora no desempenho global do sistema.

É importante lembrar que tanto a segunda quanto a terceira execuções efetuadas, simularam situações extremas e atípicas, levando em consideração o fato de executarem várias instâncias da aplicação cliente em uma mesma máquina. Esta situação foi necessária pela falta de recursos para uma simulação mais elaborada (seriam necessárias 33 máquinas para a execução da segunda bateria de testes, por exemplo, e vinte e nove pessoas executando o sistema ao mesmo tempo).

Na quarta e última execução efetuada, a única máquina utilizada como servidor de aplicação foi a D102-040, escolhida por ter uma configuração bastante robusta em relação às demais, sendo utilizado o resultado desta amostra para comparativos posteriores. As instâncias de aplicações cliente continuaram sendo executadas nas máquinas D102-043 e D102-044 e o sistema escalonador de requisições também continuou sendo executado na máquina D102-043.

Os resultados desta última execução feita em laboratório encontram-se na Tabela 5.

Tabela 5 – Tempos de execução das requisições na quarta bateria de testes

Ação	Início	Fim	Tempo Gasto
Operação de cálculo no servidor 172.16.1.216.	15:53:36	15:53:53	00:17
Operação de cálculo no servidor 172.16.1.216.	15:53:37	15:54:13	00:36
Operação de cálculo no servidor 172.16.1.216.	15:53:37	15:54:34	00:57
Operação de cálculo no servidor 172.16.1.216.	15:53:37	15:54:56	01:19
Operação de cálculo no servidor 172.16.1.216.	15:53:37	15:55:18	01:41
Operação de cálculo no servidor 172.16.1.216.	15:53:38	15:55:40	02:02
Operação de cálculo no servidor 172.16.1.216.	15:53:38	15:56:01	02:23
Operação de cálculo no servidor 172.16.1.216.	15:53:39	15:56:23	02:44
Operação de cálculo no servidor 172.16.1.216.	15:53:40	15:56:45	03:05
Operação de cálculo no servidor 172.16.1.216.	15:53:43	15:57:07	03:24
Operação de cálculo no servidor 172.16.1.216.	15:53:45	15:57:29	03:44
Operação de cálculo no servidor 172.16.1.216.	15:53:49	15:57:50	04:01
Operação de cálculo no servidor 172.16.1.216.	15:53:51	15:58:12	04:21
Operação de cálculo no servidor 172.16.1.216.	15:53:54	15:58:34	04:40
Operação de cálculo no servidor 172.16.1.216.	15:53:58	15:58:56	04:58
Operação de cálculo no servidor 172.16.1.216.	15:55:14	15:59:18	04:04
Operação de cálculo no servidor 172.16.1.216.	15:55:23	15:59:39	04:16
Operação de cálculo no servidor 172.16.1.216.	15:55:25	16:00:01	04:36
Operação de cálculo no servidor 172.16.1.216.	15:55:27	16:00:23	04:56
Operação de cálculo no servidor 172.16.1.216.	15:55:27	16:00:45	05:18
Operação de cálculo no servidor 172.16.1.216.	15:55:27	16:01:07	05:40
Operação de cálculo no servidor 172.16.1.216.	15:55:28	16:01:30	06:02
Operação de cálculo no servidor 172.16.1.216.	15:55:28	16:01:52	06:24

Observando a coluna de tempo gasto na execução das tarefas, pode-se perceber que as requisições foram enviadas a uma lista de espera, já que todas foram executadas pelo mesmo servidor. Uma tarefa começava a executar quando outra terminava, resultando num tempo de espera proporcional ao número de tarefas na fila de execução. Esta execução serve como comprovação da eficiência do escalonamento das requisições, tendo em vista que nas demais execuções mais de uma tarefa foram atendidas ao mesmo tempo, já que executavam em servidores diferentes.

As informações gravadas nos *logs* de execução dos testes mencionados podem ser visualizadas no Apêndice C.

Alguns pontos podem ser observados pela amostragem dos testes efetuados:

a) as máquinas que executaram as várias instâncias da aplicação cliente tiveram seu desempenho bastante comprometido;
b) o serviço de escalonamento de requisições utilizou recursos de processamento da máquina, porém bem menos que aqueles utilizados pelas aplicações cliente;
c) o fator utilizado para cálculo da quantidade de processos a executar em cada máquina não recebeu um ajuste correto, resultando num escalonamento desproporcional aos recursos de cada membro ativo;
d) o escalonamento de requisições contribui para a melhoria do desempenho global do sistema, porém deve ser aplicado apenas quando a demanda de pedidos para execução de tarefas tem um grande volume;
e) durante a última execução, onde foi utilizado um único servidor de aplicação, o serviço de RPC do Windows falhou quando um grande volume de requisições foi enviado ao mesmo tempo, situação esta que não pôde ser observada durante o uso de várias máquinas fazendo parte do processo de escalonamento.

4 CONCLUSÕES

Com a observação dos resultados obtidos nos testes feitos em laboratório, concluiu-se que a tarefa principal de melhorar o desempenho global do sistema teve um resultado positivo, mesmo sem um ajuste perfeito da equação que determina o percentual de ociosidade de cada um dos servidores cadastrados. Com um ajuste fino dos parâmetros envolvidos na equação, conseguido com a mudança do fator de cálculo da quantidade de processos a executar em cada máquina (variável que pode ser ajustada pelo usuário no configurador do sistema), o rendimento pode ser ainda melhor e os benefícios do escalonamento tornam-se ainda mais evidentes.

O sistema conseguiu atender a premissa básica de escalonar as requisições de tarefas de diversas aplicações clientes entre os membros do grupo que estavam aptos à atender os pedidos efetuados, ou melhor, que tinham as aplicações servidoras instaladas e operacionais.

Em relação ao objetivo inicial da confecção de um escalonador que apenas designasse a melhor máquina para atender a requisição, não tendo responsabilidade por encaminhar a requisição ao servidor propriamente dito, o resultado alcançado foi bastante satisfatório, já que o custo de comunicação foi relativamente baixo e a máquina responsável pelo escalonamento das requisições não teve seu desempenho prejudicado pela sobrecarga de tarefas a executar.

Por outro lado, é possível que o escalonamento tivesse um melhor aproveitamento se fosse feito de forma dinâmica, baseado nos tempos gastos por cada computador no atendimento das tarefas, porém, como essa técnica não foi implementada e testada, isso se torna uma especulação.

De qualquer forma, poupar o escalonador da tarefa de encaminhar as requisições (ou solicitar a execução dos processos) teve um resultado bastante significativo, demonstrando a capacidade de se atender uma maior quantidade de clientes sem uma perda muito grande de *performance*.

Em relação ao aprendizado obtido com a elaboração deste trabalho, é importante enfatizar os seguintes pontos:

a) utilização das tecnologias COM/DCOM de uma maneira diferente da trivial, informando um caminho dinamicamente para as requisições ao servidor de aplicação;

b) a importância e benefícios da utilização de múltiplos computadores na execução de tarefas comuns de sistemas de controle administrativo;
c) o custo de desempenho gerado pela interação simples do usuário com esse tipo de sistema, através das aplicações *front-end*;
d) o alto poder de processamento conseguido até mesmo com máquinas aparentemente ultrapassadas e com configurações modestas;
e) a diferença de desempenho conseguida com processadores novos e mais robustos, e como essa diferença torna-se bem menos visível quando são efetuadas operações de I/O (entrada e saída de dados), como leitura e gravação de arquivos em disco rígido.

4.1 EXTENSÕES

Como o sistema desenvolvido utiliza um algoritmo de escalonamento determinístico (estático), uma das propostas para implementações futuras, seria utilizar os dados de tempo de processamento de cada requisição (já mantidos pelo sistema) para o desenvolvimento de um algoritmo dinâmico, onde a decisão de escolha do servidor fosse baseada nas informações do tempo gasto para o atendimento da tarefa por cada uma das máquinas onde as aplicações servidoras encontram-se instaladas. Desta forma, o escalonador deve efetuar um cálculo da velocidade de atendimento de cada máquina baseado no tempo gasto por ela para atender determinada tarefa, obtendo uma maior precisão à nível de processo requisitado, pois cada tarefa tem um custo diferente para os computadores que as executam.

Outra implementação possível seria a de que o próprio ambiente de escalonamento ficasse encarregado de requisitar as tarefas aos servidores e encaminhar o resultado obtido nas operações aos clientes que efetuaram o pedido. Este tipo de algoritmo já é praticado por alguns tipos de escalonadores, porém como vislumbrado em seção específica da fundamentação teórica, é bastante comum que servidor de escalonamento fique sobrecarregado com estes processos. Uma característica contida no sistema que ameniza bastante esta questão é o fato do escalonador não fazer parte do grupo de máquinas que contém as aplicações servidoras instaladas. Mas mesmo assim, para que a implementação destes recursos se torne viável, faz-se necessário que este problema de sobrecarga seja sanado e que o tempo gasto por todo este processo seja minimizado.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORIM, Cláudio Luis de; BARBOSA, Valmir Carneiro; FERNANDES, Edil Seberiano Tavares. **Uma introdução à computação paralela e distribuída**. Campinas: Editora da Unicamp, IMECC, 1988.

BERSON, Alex. **Client/Server Architecture** – Second Edition. New York: McGraw-Hill Book Companies, 1996.

BODART, N.L.O. et al. Balanceamento de carga nas redes inteligentes distribuídas. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 5., 2004, Foz do Iguaçu. **Anais...** Foz do Iguaçu, 2004. p. 28-35.

BRANCO, Kalinka Regina Lucas Jaquie Castelo. **Índices de carga e desempenho em ambientes paralelos/distribuídos**: modelagem e métricas. 2004. 232 f. Tese (Doutorado em Ciências) – Ciências de Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Paulo.

EDDON, Guy; EDDON, Henry. **Inside distributed COM**. Washington: Microsoft Press, 1998.

ELASSER, Robert; MONIEN, Burkhard; PREIS, Robert. Diffusive load balancing schemes on heterogeneous networks. In: ACM SYMPOSIUM ON PARALLEL ALGORITHMS AND ARCHITECTURES, 12, 2000, Bar Harbor. **Proceedings…**USA: IEEE, 2000. p.30-38.

HORSTMANN, Markus; KIRTLAND, Mary. **DCOM architecture**. New York: Microsoft Press, 1997.

LAGES, Newton Alberto de Castilho; NOGUEIRA, José Marcos Silva. **Introdução aos sistemas distribuídos**. Campinas: Editora da Unicamp, 1986.

MICROSOFT. **Microsoft Security Bulletin**. [S.I.], 2003. Disponível em: <http://www.microsoft.com/brasil/technet/Boletins/BoletinsMS03_26.aspx>. Acesso em: 25 mar. 2006.

MICROSOFT. **Modelo de Definição de Sistemas**. [S.I.], 2005. Disponível em: <http://www.microsoft.com/portugal/windowsserversystem/dsi/sdm.mspx>. Acesso em: 25 mar. 2006.

NOGUEIRA, M. L. B. ; YAMIN, A. C. ; VARGAS, Patrícia Kayser ; GEYER, Cláudio Fernando Resin . Balanceamento de carga em sistemas distribuídos: uma proposta de ambiente para avaliação. In: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA, 2001, **Anais**... Mérida, Venezuela, 2001.

PINTO, A. R.; DANTAS, M. A. R. Servidor Genético: uma abordagem de balanceamento de carga baseada em algoritmo de aprendizado de máquina genético para agregados de computadores. **Journal of Computer Science**, [S.l.], v. 5, n. 1, p. 51-60, Mar. 2006.

PITANGA, Marcos. **Computação em cluster: o estado da arte da computação**. Rio de Janeiro: Brasport, 2003

QUEIROZ, José A. M. de; CUNHA, Paulo R. F. **Sistemas distribuídos**: de especificações LOTOS a implementações. Recife: Universidade Federal de Pernambuco, 1994.

SCHLEMER, Elgio; GEYER, Fernando Resin. **Escalonamento em sistemas distribuídos**: uma introdução. Porto Alegre, 1998. Disponível em: <http://www.inf.ufrgs.br/~elgios/trabs-html/escalona/EscProc.html>. Acesso em: 18 mar. 2006.

SILVA, Alex de Araújo; GOMIDE, Carlos Francisco; PETRILLO, Fábio. **Metodologia e projeto de software orientados a objetos:** modelando, projetando e desenvolvendo sistemas com UML e componentes distribuídos. São Paulo: Editora Érica Ltda, 2003.

SPINDOLA, T. ; WESTPHALL, C. B. ; KOCH, F. L. Balanceamento de carga em grids de agentes para gerência de redes de computadores. In: XXII Simpósio Brasileiro de Redes de Computadores, 2004, **Anais**... Gramado (RS), 2004.

STRACK, Jair. **Sistemas de processamento distribuído**. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 1984.

APÊNDICE A – Detalhamentos dos casos de uso

Este apêndice contém o detalhamento e uma breve explicação de todos os casos de uso descritos na seção destinada à especificação do software.

Identificação do UC	UC01.01 Preenche dados dos computadores
Objetivo	Permite ao usuário informar os dados de localização e configuração dos servidores de aplicação para fazerem parte do grupo disponível para escalonamento. No cadastramento dos computadores é necessário informar qual o tipo de aplicação atendida por ele, já que será possível efetuar o escalonamento de servidores contendo diversas atividades diferentes, além das informações de configuração física dos equipamentos, um fator para cálculo do número de processos a executar pela aplicação e seu endereço IP, utilizado na requisição das tarefas. Será permitido ao usuário desativar um servidor através da lista de consulta e alterar os dados cadastrados previamente, porém os servidores não podem ser excluídos. (Figuras 20 e 21)

Identificação do UC	UC02.01 Cadastra computadores
Objetivo	A partir dos dados enviados pelo configurador do sistema contendo as informações das máquinas pertencentes ao grupo de escalonamento, o escalonador grava um arquivo onde estas são guardadas para utilização futura. Este arquivo além de utilizado para a escolha do servidor com maior disponibilidade, é acessado ainda para a montagem da lista de computadores que é visualizada no componente de cadastro de servidores. (Figuras 20 e 21)

Identificação do UC	UC02.02 Encontra o servidor com maior disponibilidade
Objetivo	Através de uma requisição efetuada pela aplicação cliente ao escalonador de processos, é verificado quais os servidores aptos para atender a operação e efetuado um cálculo para determinar qual servidor encontra-se com níveis de carga de processamento menor, ou seja, em melhores condições de atender a requisição no momento. Depois de eleito o servidor adequado, é montado um registro contendo os dados de localização do mesmo e enviado para a aplicação cliente, que se encarregará de fazer a comunicação diretamente com a aplicação servidora.

Identificação do UC	UC02.03 Retira processo da fila de processamento
Objetivo	Após a conclusão da tarefa por parte do servidor de aplicação, o cliente informa ao escalonador que o processo terminou de executar, bem como o tempo gasto na execução do mesmo. Esta função é indispensável para que as informações de carga de processamento dos computadores estejam sempre atualizadas, sendo que o escalonador retira o processo em questão da lista de carga de processamento pertencente ao computador selecionado.

Identificação do UC	UC02.04 Busca informações do log
Objetivo	As informações de log são mantidas pelo escalonador, que grava um arquivo texto contendo detalhadamente os passos seguidos para a tomada das decisões. Sempre que o usuário solicita a visualização do log, o sistema efetua a chamada de um método no escalonador que busca as informações atualizadas deste arquivo para exibi-las na tela contida na aplicação cliente. (Figura 23)

.Identificação do UC	UC03.01 Solicita execução de processo
Objetivo	Após a seleção do servidor com maior disponibilidade, a aplicação cliente (Figura 22) solicita a execução do método desejado que se encontra disponível na aplicação servidora. Durante este processo, a aplicação servidora calcula o tempo gasto na execução da operação e envia esta informação à aplicação cliente.

.Identificação do UC	UC03.02 Visualiza log de execução
Objetivo	O sistema disponibiliza ao usuário um log dos passos que o escalonador segue para a escolha dos computadores que atendem as requisições efetuadas, bem como os pedidos e o término dessas tarefas. Para tanto, existe um componente que permite a visualização dessas informações (Figura 23) de maneira simples e objetiva, trazendo o horário em que cada função foi executada.

Identificação do UC	UC04.01 Executa processo
Objetivo	Efetua o processamento das informações necessárias para o atendimento da requisição efetuada pela aplicação cliente, preocupando-se ainda em informar o tempo gasto na execução do processo, informação que será encaminhada ao cliente e posteriormente ao escalonador de requisições.

APÊNDICE B – Gráficos dos tempos médios de atendimento das requisições por máquina em cada uma das execuções

Este apêndice contém as informações de tempo médio de atendimento das requisições em cada uma das execuções efetuadas para teste do sistema no laboratório.

O gráfico 1 demonstra os tempos médios de atendimento alcançados em cada uma das execuções.

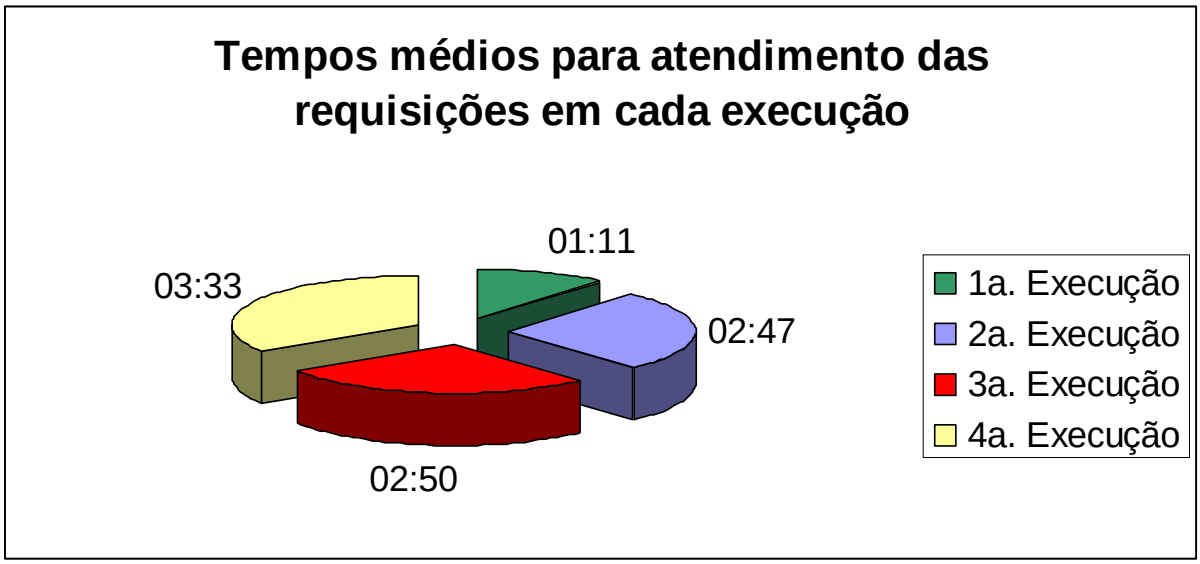

Gráfico 1 – Tempos médios para atendimento das requisições em cada execução

Faz-se necessário ressaltar a quantidade de requisições efetuadas em cada uma das execuções, sendo que não houve uma homogeneidade nesse sentido em relação às baterias de testes. Desta forma, destaca-se que na primeira exeução, foram efetuadas 10 requisições. Na segunda esse número passou para 29 requisições, numa tentativa de explorar o sistema mais intensamente, permanecendo em 27 requisições na terceira fase e 23 na quarta e última, onde se utilizou apenas a máquina com recursos mais poderosos de processamento.

É possível ainda visualizar no gráfico 2, os tempos divididos por máquina para uma análise mais aprofundada dos resultados obtidos.

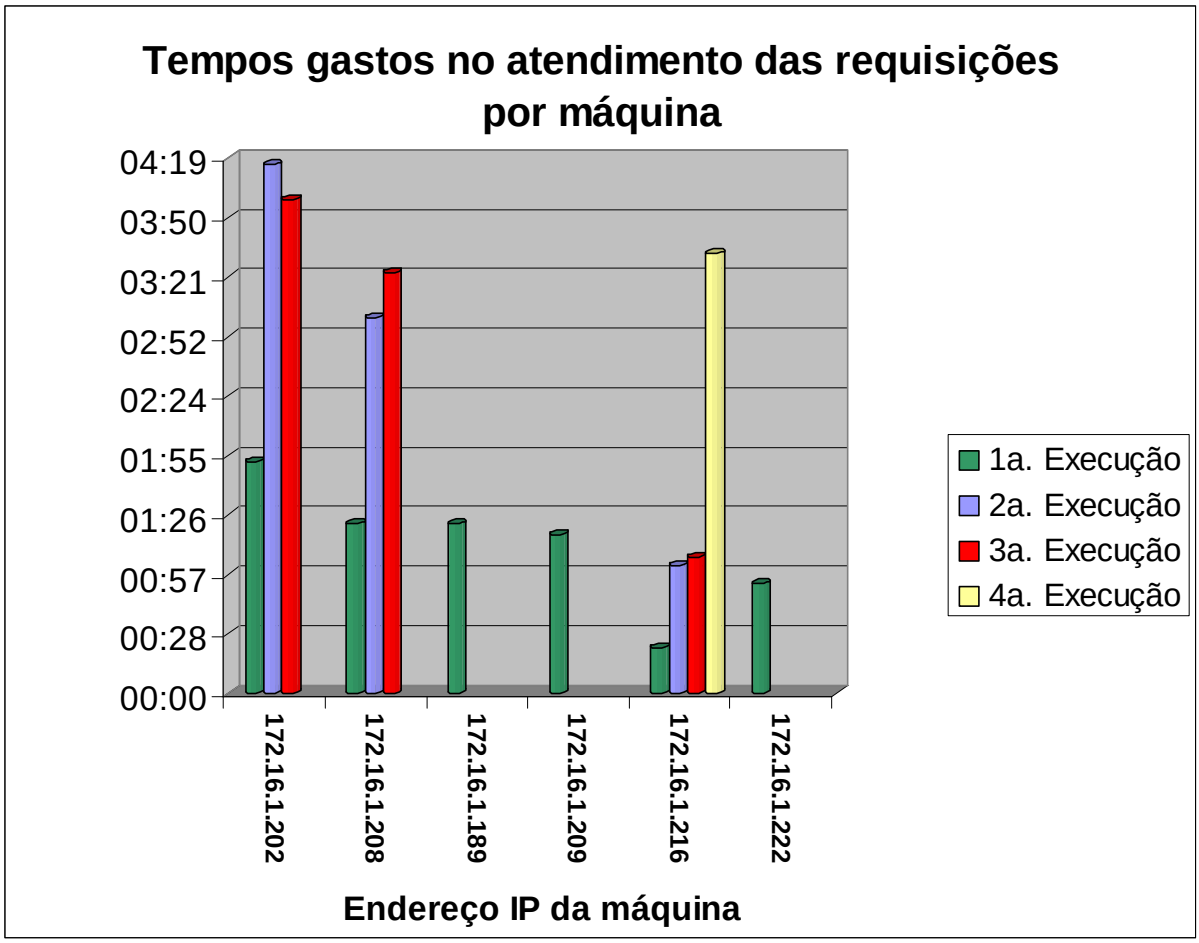

Gráfico 2 – Tempos médios gastos no atendimento das requisições por máquina

APÊNDICE C – Logs de execução do sistema de escalonamento nos testes efetuados em laboratório

Este apêndice contém as informações gravadas no *log* do sistema de escalonamento, onde podem ser encontradas em detalhes as operações efetuadas durante a execução dos testes.

Abaixo pode ser visualizado o texto completo contido no arquivo de *log* da primeira bateria de testes efetuada, utilizando-se de todas as máquinas disponíveis:

15:08:52 - Requisitada tarefa de Calculo.
15:08:52 - Servidor de Calculo (172.16.1.202), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.208), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.189), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.209), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:52 - Escolhido servidor de Calculo: 172.16.1.202.
15:08:52 - Requisitada tarefa de Calculo.
15:08:52 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.208), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.189), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.209), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:08:52 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:52 - Escolhido servidor de Calculo: 172.16.1.208.
15:08:53 - Requisitada tarefa de Calculo.
15:08:53 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:53 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:53 - Servidor de Calculo (172.16.1.189), está com 100 % de seus recursos disponíveis.
15:08:53 - Servidor de Calculo (172.16.1.209), está com 100 % de seus recursos disponíveis.
15:08:53 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:08:53 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:53 - Escolhido servidor de Calculo: 172.16.1.189.
15:08:54 - Requisitada tarefa de Calculo.
15:08:54 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:54 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:54 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:54 - Servidor de Calculo (172.16.1.209), está com 100 % de seus recursos disponíveis.
15:08:54 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:08:54 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:54 - Escolhido servidor de Calculo: 172.16.1.209.
15:08:55 - Requisitada tarefa de Calculo.
15:08:55 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:55 - Escolhido servidor de Calculo: 172.16.1.216.
15:08:55 - Requisitada tarefa de Calculo.
15:08:55 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.

15:08:55 - Servidor de Calculo (172.16.1.216), está com 98 % de seus recursos disponíveis.
15:08:55 - Servidor de Calculo (172.16.1.222), está com 100 % de seus recursos disponíveis.
15:08:55 - Escolhido servidor de Calculo: 172.16.1.222.
15:08:57 - Requisitada tarefa de Calculo.
15:08:57 - Servidor de Calculo (172.16.1.202), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.216), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.222), está com 98 % de seus recursos disponíveis.
15:08:57 - Escolhido servidor de Calculo: 172.16.1.202.
15:08:57 - Requisitada tarefa de Calculo.
15:08:57 - Servidor de Calculo (172.16.1.202), está com 96 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.208), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.216), está com 98 % de seus recursos disponíveis.
15:08:57 - Servidor de Calculo (172.16.1.222), está com 98 % de seus recursos disponíveis.
15:08:57 - Escolhido servidor de Calculo: 172.16.1.208.
15:08:58 - Requisitada tarefa de Calculo.
15:08:58 - Servidor de Calculo (172.16.1.202), está com 96 % de seus recursos disponíveis.
15:08:58 - Servidor de Calculo (172.16.1.208), está com 96 % de seus recursos disponíveis.
15:08:58 - Servidor de Calculo (172.16.1.189), está com 98 % de seus recursos disponíveis.
15:08:58 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.
15:08:58 - Servidor de Calculo (172.16.1.216), está com 98 % de seus recursos disponíveis.
15:08:58 - Servidor de Calculo (172.16.1.222), está com 98 % de seus recursos disponíveis.
15:08:58 - Escolhido servidor de Calculo: 172.16.1.189.
15:08:59 - Requisitada tarefa de Calculo.
15:08:59 - Servidor de Calculo (172.16.1.202), está com 96 % de seus recursos disponíveis.
15:08:59 - Servidor de Calculo (172.16.1.208), está com 96 % de seus recursos disponíveis.
15:08:59 - Servidor de Calculo (172.16.1.189), está com 96 % de seus recursos disponíveis.
15:08:59 - Servidor de Calculo (172.16.1.209), está com 98 % de seus recursos disponíveis.
15:08:59 - Servidor de Calculo (172.16.1.216), está com 98 % de seus recursos disponíveis.
15:08:59 - Servidor de Calculo (172.16.1.222), está com 98 % de seus recursos disponíveis.
15:08:59 - Escolhido servidor de Calculo: 172.16.1.209.
15:09:17 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:09:47 - Operação de Calculo terminou a execução no servidor 172.16.1.209.
15:09:48 - Operação de Calculo terminou a execução no servidor 172.16.1.222.
15:09:49 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:09:49 - Operação de Calculo terminou a execução no servidor 172.16.1.189.
15:09:51 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:10:40 - Operação de Calculo terminou a execução no servidor 172.16.1.209.
15:10:44 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:10:45 - Operação de Calculo terminou a execução no servidor 172.16.1.189.
15:11:41 - Operação de Calculo terminou a execução no servidor 172.16.1.202.

A seguir está listado o texto do arquivo de *log* completo da terceira execução efetuada, contando com três máquinas executando a aplicação servidora:

15:42:21 - Requisitada tarefa de Calculo.
15:42:21 - Servidor de Calculo (172.16.1.202), está com 100 % de seus recursos disponíveis.
15:42:21 - Servidor de Calculo (172.16.1.208), está com 95 % de seus recursos disponíveis.
15:42:21 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:42:21 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:25 - Requisitada tarefa de Calculo.
15:42:25 - Servidor de Calculo (172.16.1.202), está com 95 % de seus recursos disponíveis.
15:42:25 - Servidor de Calculo (172.16.1.208), está com 95 % de seus recursos disponíveis.
15:42:25 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:42:25 - Escolhido servidor de Calculo: 172.16.1.216.

15:42:26 - Requisitada tarefa de Calculo.
15:42:26 - Servidor de Calculo (172.16.1.202), está com 95 % de seus recursos disponíveis.
15:42:26 - Servidor de Calculo (172.16.1.208), está com 95 % de seus recursos disponíveis.
15:42:26 - Servidor de Calculo (172.16.1.216), está com 95 % de seus recursos disponíveis.
15:42:26 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:31 - Requisitada tarefa de Calculo.
15:42:31 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:42:31 - Servidor de Calculo (172.16.1.208), está com 95 % de seus recursos disponíveis.
15:42:31 - Servidor de Calculo (172.16.1.216), está com 95 % de seus recursos disponíveis.
15:42:31 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:32 - Requisitada tarefa de Calculo.
15:42:32 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:42:32 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:42:32 - Servidor de Calculo (172.16.1.216), está com 95 % de seus recursos disponíveis.
15:42:32 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:33 - Requisitada tarefa de Calculo.
15:42:33 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.216), está com 91 % de seus recursos disponíveis.
15:42:33 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:33 - Requisitada tarefa de Calculo.
15:42:33 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.216), está com 86 % de seus recursos disponíveis.
15:42:33 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:33 - Requisitada tarefa de Calculo.
15:42:33 - Servidor de Calculo (172.16.1.202), está com 85 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:42:33 - Servidor de Calculo (172.16.1.216), está com 86 % de seus recursos disponíveis.
15:42:33 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:34 - Requisitada tarefa de Calculo.
15:42:34 - Servidor de Calculo (172.16.1.202), está com 85 % de seus recursos disponíveis.
15:42:34 - Servidor de Calculo (172.16.1.208), está com 85 % de seus recursos disponíveis.
15:42:34 - Servidor de Calculo (172.16.1.216), está com 86 % de seus recursos disponíveis.
15:42:34 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:34 - Requisitada tarefa de Calculo.
15:42:34 - Servidor de Calculo (172.16.1.202), está com 85 % de seus recursos disponíveis.
15:42:34 - Servidor de Calculo (172.16.1.208), está com 85 % de seus recursos disponíveis.
15:42:34 - Servidor de Calculo (172.16.1.216), está com 82 % de seus recursos disponíveis.
15:42:34 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:34 - Requisitada tarefa de Calculo.
15:42:35 - Servidor de Calculo (172.16.1.202), está com 80 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.208), está com 85 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.216), está com 82 % de seus recursos disponíveis.
15:42:35 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:35 - Requisitada tarefa de Calculo.
15:42:35 - Servidor de Calculo (172.16.1.202), está com 80 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.208), está com 80 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.216), está com 82 % de seus recursos disponíveis.
15:42:35 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:35 - Requisitada tarefa de Calculo.
15:42:35 - Servidor de Calculo (172.16.1.202), está com 80 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.208), está com 80 % de seus recursos disponíveis.
15:42:35 - Servidor de Calculo (172.16.1.216), está com 78 % de seus recursos disponíveis.
15:42:35 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:36 - Requisitada tarefa de Calculo.
15:42:36 - Servidor de Calculo (172.16.1.202), está com 76 % de seus recursos disponíveis.
15:42:36 - Servidor de Calculo (172.16.1.208), está com 80 % de seus recursos disponíveis.
15:42:36 - Requisitada tarefa de Calculo.
15:42:36 - Servidor de Calculo (172.16.1.202), está com 76 % de seus recursos disponíveis.

15:42:36 - Servidor de Calculo (172.16.1.208), está com 80 % de seus recursos disponíveis.
15:42:36 - Servidor de Calculo (172.16.1.216), está com 78 % de seus recursos disponíveis.
15:42:36 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:36 - Servidor de Calculo (172.16.1.216), está com 78 % de seus recursos disponíveis.
15:42:36 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:36 - Requisitada tarefa de Calculo.
15:42:36 - Servidor de Calculo (172.16.1.202), está com 76 % de seus recursos disponíveis.
15:42:36 - Servidor de Calculo (172.16.1.208), está com 76 % de seus recursos disponíveis.
15:42:36 - Servidor de Calculo (172.16.1.216), está com 78 % de seus recursos disponíveis.
15:42:36 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:37 - Requisitada tarefa de Calculo.
15:42:37 - Servidor de Calculo (172.16.1.202), está com 76 % de seus recursos disponíveis.
15:42:37 - Servidor de Calculo (172.16.1.208), está com 76 % de seus recursos disponíveis.
15:42:37 - Servidor de Calculo (172.16.1.216), está com 73 % de seus recursos disponíveis.
15:42:37 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:37 - Requisitada tarefa de Calculo.
15:42:37 - Servidor de Calculo (172.16.1.202), está com 71 % de seus recursos disponíveis.
15:42:37 - Servidor de Calculo (172.16.1.208), está com 76 % de seus recursos disponíveis.
15:42:37 - Servidor de Calculo (172.16.1.216), está com 73 % de seus recursos disponíveis.
15:42:37 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:38 - Requisitada tarefa de Calculo.
15:42:38 - Servidor de Calculo (172.16.1.202), está com 71 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.208), está com 71 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.216), está com 73 % de seus recursos disponíveis.
15:42:38 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:38 - Requisitada tarefa de Calculo.
15:42:38 - Servidor de Calculo (172.16.1.202), está com 71 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.208), está com 71 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.216), está com 69 % de seus recursos disponíveis.
15:42:38 - Escolhido servidor de Calculo: 172.16.1.202.
15:42:38 - Requisitada tarefa de Calculo.
15:42:38 - Servidor de Calculo (172.16.1.202), está com 66 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.208), está com 71 % de seus recursos disponíveis.
15:42:38 - Servidor de Calculo (172.16.1.216), está com 69 % de seus recursos disponíveis.
15:42:38 - Escolhido servidor de Calculo: 172.16.1.208.
15:42:39 - Requisitada tarefa de Calculo.
15:42:39 - Servidor de Calculo (172.16.1.202), está com 66 % de seus recursos disponíveis.
15:42:39 - Servidor de Calculo (172.16.1.208), está com 66 % de seus recursos disponíveis.
15:42:39 - Servidor de Calculo (172.16.1.216), está com 69 % de seus recursos disponíveis.
15:42:39 - Escolhido servidor de Calculo: 172.16.1.216.
15:42:47 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:43:09 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:43:28 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:43:30 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:43:52 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:44:14 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:44:24 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:44:36 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:44:58 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:44:58 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:45:19 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:45:19 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:45:32 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:45:34 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:46:15 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:46:21 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:47:11 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:47:19 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:47:49 - Requisitada tarefa de Calculo.
15:47:49 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.

15:47:49 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:47:49 - Servidor de Calculo (172.16.1.216), está com 100 % de seus recursos disponíveis.
15:47:49 - Escolhido servidor de Calculo: 172.16.1.216.
15:47:51 - Requisitada tarefa de Calculo.
15:47:51 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:47:51 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:47:51 - Servidor de Calculo (172.16.1.216), está com 95 % de seus recursos disponíveis.
15:47:51 - Escolhido servidor de Calculo: 172.16.1.216.
15:47:52 - Requisitada tarefa de Calculo.
15:47:52 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:47:52 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:47:52 - Servidor de Calculo (172.16.1.216), está com 91 % de seus recursos disponíveis.
15:47:52 - Escolhido servidor de Calculo: 172.16.1.216.
15:47:57 - Requisitada tarefa de Calculo.
15:47:57 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:47:57 - Servidor de Calculo (172.16.1.208), está com 90 % de seus recursos disponíveis.
15:47:57 - Servidor de Calculo (172.16.1.216), está com 86 % de seus recursos disponíveis.
15:47:57 - Escolhido servidor de Calculo: 172.16.1.202.
15:48:06 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:48:11 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:48:17 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:48:27 - Requisitada tarefa de Calculo.
15:48:27 - Servidor de Calculo (172.16.1.202), está com 90 % de seus recursos disponíveis.
15:48:27 - Servidor de Calculo (172.16.1.208), está com 95 % de seus recursos disponíveis.
15:48:27 - Servidor de Calculo (172.16.1.216), está com 91 % de seus recursos disponíveis.
15:48:28 - Escolhido servidor de Calculo: 172.16.1.208.
15:48:33 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:48:55 - Operação de Calculo terminou a execução no servidor 172.16.1.216.
15:49:02 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:49:16 - Operação de Calculo terminou a execução no servidor 172.16.1.202.
15:49:58 - Operação de Calculo terminou a execução no servidor 172.16.1.208.
15:50:19 - Operação de Calculo terminou a execução no servidor 172.16.1.202.