1
Introdução

Recentes avanços em redes de computadores impulsionaram a busca e o desenvolvimento de meios para facilitar e acelerar o desenvolvimento de aplicações em sistemas distribuídos, tornando possível novas formas de se estruturar aplicações e sistemas computacionais, de maneira a aproveitar os benefícios desses ambientes distribuídos. Em particular, a possibilidade de particionar os diversos elementos funcionais de uma aplicação [5], de forma que cada um seja executado, em paralelo, em um nó de computação diferente, representa uma solução efetiva e barata para se obter um melhor desempenho (Figura 1). Pode-se ainda citar benefícios como compartilhamento de recursos, maior disponibilidade dos serviços da aplicação, potencial para tolerância à falhas, e melhor escalabilidade.

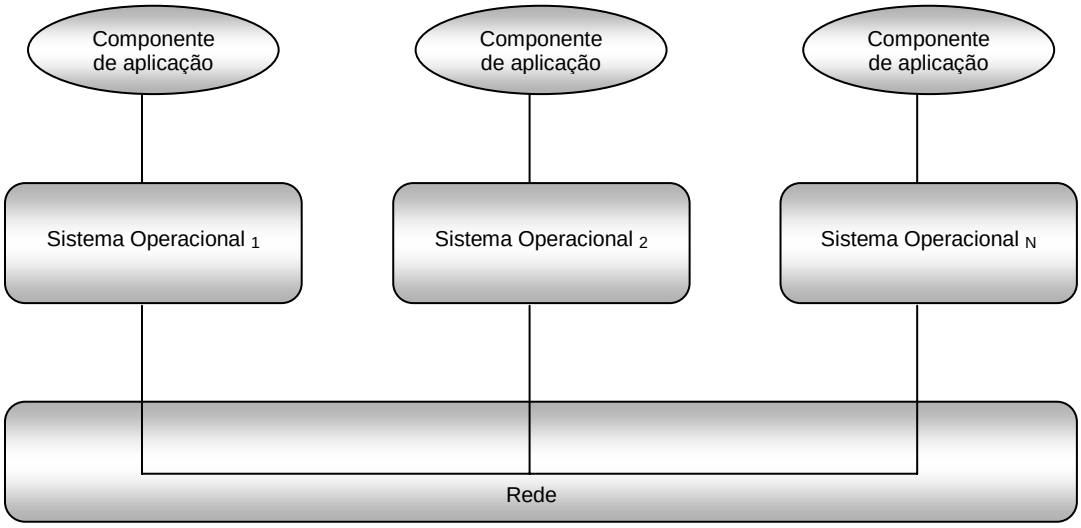

Figura 1 - Elementos funcionais de uma aplicação sendo executados de forma distribuída

Entretanto, o desenvolvimento de aplicações em ambientes distribuídos apresenta complicações não existentes em sistemas centralizados. Isto se deve,

principalmente, ao fato de que diversos componentes das aplicações encontram-se dispersos em nós de computação independentes, de forma que a interação entre os mesmos requer comunicação através da rede. Ainda como conseqüência desta distribuição, outras questões importantes devem ser consideradas no desenvolvimento de aplicações, tais como heterogeneidade nos níveis de *hardware*, sistemas operacionais e linguagens de programação, tolerância à falhas (que falhas em um determinado nó não necessariamente afetem os demais), concorrência, segurança, e latência de comunicação. Na ausência de uma infra-estrutura de suporte adequada, tais questões tendem a tornar o projeto e implementação de aplicações distribuídas uma tarefa de complexidade extrema, limitando seu uso em situações reais [16].

Frameworks desenvolvidos com propósitos específicos, tais como para abstração no acesso a aglomerados de computadores (*cluster computing*) [15], e tecnologias de *middleware* para suporte a comunicação [12] foram propostos com o objetivo de fornecer a infra-estrutura necessária para facilitar o desenvolvimento de aplicações distribuídas. Esses *frameworks* e *middlewares* (Figura 2) procuram esconder do programador os detalhes da comunicação na rede e as diferenças entre as plataformas de *hardware* e *software* em sistemas heterogêneos. Trata-se de uma camada de *software*, residente acima do sistema operacional e da camada de comunicação, que oferece abstrações de alto nível, com objetivo de facilitar a programação distribuída. As abstrações oferecidas tentam fornecer uma visão uniforme na utilização de recursos heterogêneos existentes nas camadas de sistema operacional e rede.

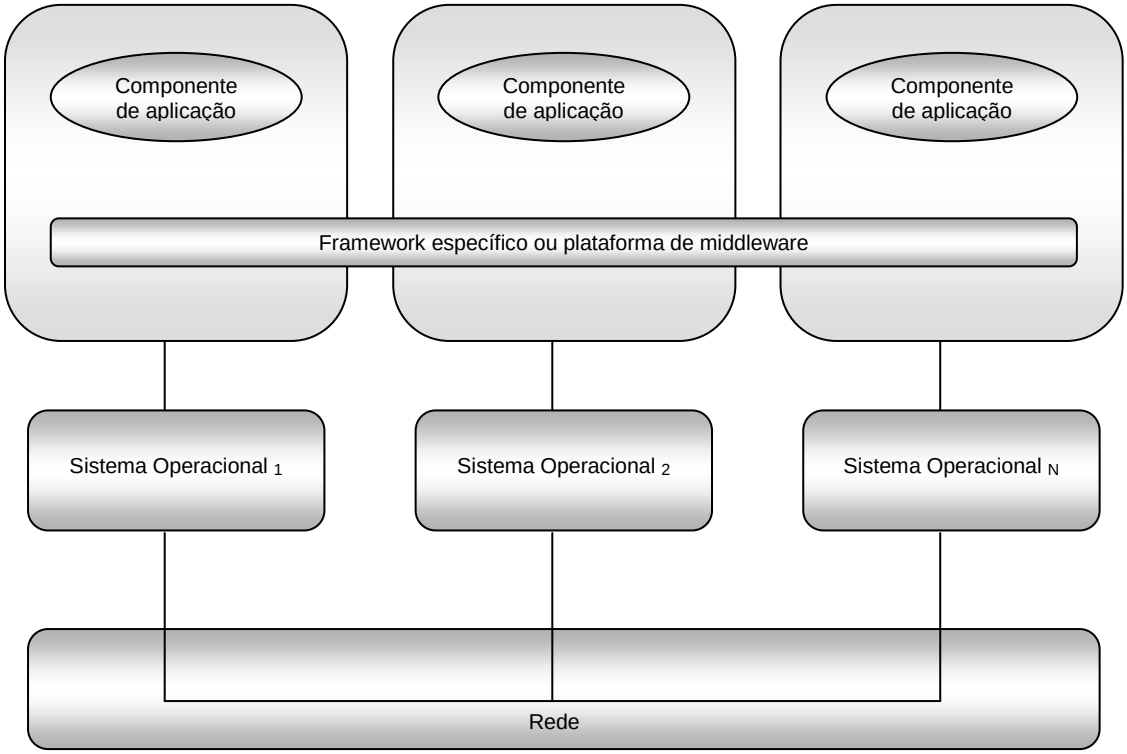

Figura 2 - Integração proporcionada por um *framework* ou um *middleware*

Em sua forma mais básica, um *framework* específico ou uma plataforma de *middleware* permite que a comunicação entre os diversos componentes no mesmo nível da aplicação seja realizada de maneira transparente do ponto de vista do desenvolvedor de aplicações. Aspectos como comunicação remota, diferentes modos de acesso e diferentes formatos de dados são escondidos do programador. Em tal cenário, o desenvolvimento de aplicações distribuídas tende, idealmente, a assumir um grau de complexidade semelhante ao caso de aplicações centralizadas, uma vez que o desenvolvedor precisa se concentrar apenas nos problemas específicos da aplicação ou propriedades funcionais dessa aplicação – problemas oriundos de distribuição são tratados transparentemente pela plataforma.

A ampla disseminação de *frameworks* específicos e de tecnologias de *middlewares* além dos recentes avanços obtidos nessa área, têm favorecido a adoção de arquiteturas distribuídas para aplicações e serviços, como em [15] [27]. Destacam-se os modelos de arquiteturas cliente-servidor (Figura 3), ponto-a-ponto, e serviços *web*, cada um apropriado para uma determinada categoria de aplicações, tais como sistemas de informações organizacionais,

disseminação e troca de informações e comércio eletrônico. Em decorrência da adoção destes tipos de arquiteturas – e das tecnologias subjacentes – ocorreu um considerável amadurecimento nessa área e, conseqüentemente, uma crescente sofisticação das aplicações distribuídas existentes. Ocorreu também o surgimento de novas categorias de aplicações explorando as potencialidades dos sistemas distribuídos, tais como aplicações distribuídas de alto desempenho, aplicações multimídia distribuídas e aplicações móveis [17]. Para essas novas aplicações, houve um aprimoramento nos requisitos em relação ao suporte oferecido pelos *frameworks* ou pelos *middlewares*.

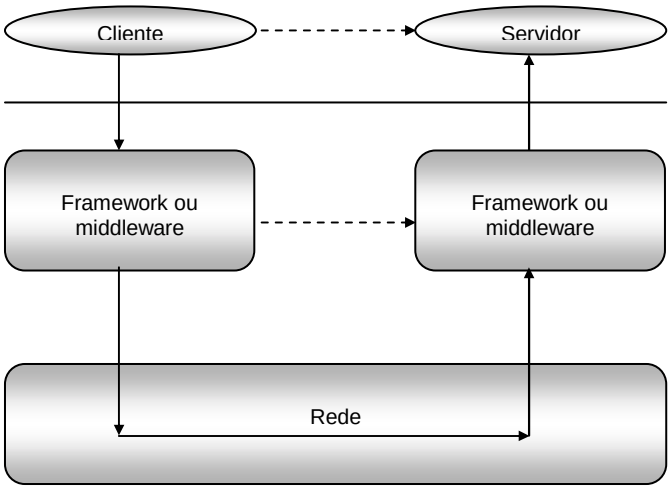

Figura 3 - Comunicação para a arquitetura cliente/servidor

Em muitos desses novos domínios de aplicações existe a necessidade de garantias de tempo nas redes, nos sistemas operacionais, e nos componentes do *framework* ou do *middleware*, para satisfazer a um novo requisito: a qualidade de serviço[1] (QoS - *Quality of Service*) [27]. Nos últimos anos, o conceito de qualidade de serviço tem sido largamente empregado em domínios como aplicações multimídia e de tempo real, onde é tipicamente associado aos

[1] Totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas, envolvidas, explícitas ou implícitas [ISO9126 - 1994].

requisitos impostos por esse tipo de aplicação a suas infra-estruturas de processamento e de comunicação [15]. Esse conceito, entretanto, pode ser empregado num sentido mais amplo, englobando qualquer característica, ou propriedade, apresentada pelo ambiente de execução de uma aplicação [13] [14].

Essa nova demanda de requisitos fez com que, aplicações no domínio específico de sistemas dinâmicos e heterogêneos (Figura 4) como a internet, redes sem fio e ambientes de computação móvel e ubíqua – onde grandes flutuações no ambiente de execução causam grandes alterações na qualidade de serviço (QoS) ofertada – provoquem um desbalanceamento na reserva e gerência dos recursos disponíveis, devido, principalmente, a ausência de mecanismos tão dinâmicos quanto sejam as necessidades do ambiente.

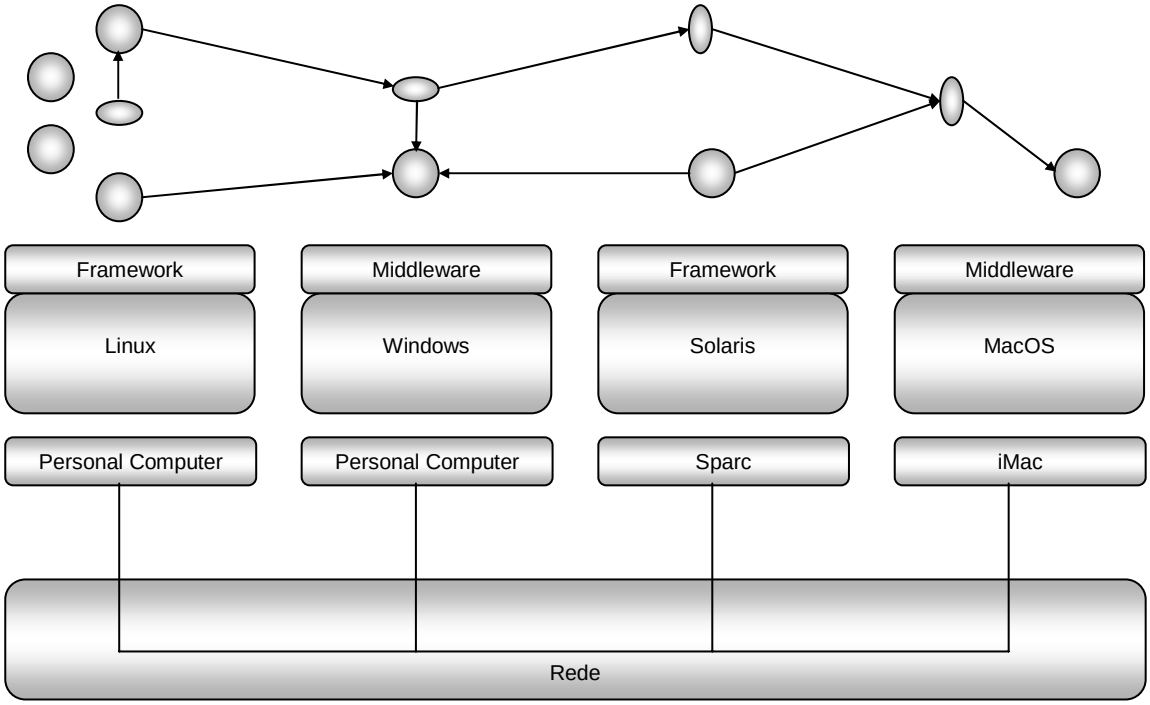

Figura 4 - Interações entre aplicações de sistemas dinâmicos e heterogêneos

No entanto, apesar de todas as pesquisas na área de sistemas distribuídos, a reserva balanceada e a gerência dos recursos desses sistemas, de seus respectivos sistemas operacionais e das redes utilizadas, são quase

sempre utilizados de uma forma simplista ou ineficiente. Alguns poucos serviços – como por exemplo, sistemas de arquivos e a *web* – são distribuídos através de um modelo cliente/servidor onde os servidores são quase sempre fixos e utilizados de forma sazonal. A maior parte das aplicações (como editores de texto, calendários, planilhas, navegadores da *web*) são regularmente executadas localmente na máquina do cliente, mesmo que esta plataforma não seja a mais apropriada para isso e mesmo que haja uma enorme quantidade de recursos disponíveis na rede local para executar essas aplicações de forma mais eficiente. Como conseqüência, os recursos ficam muitas vezes ociosos ao mesmo tempo em que os usuários não obtêm uma qualidade de serviço (QoS) satisfatória.

1.1. Ambiente proposto

Neste trabalho, busca-se resgatar o conceito de gerência dos recursos (processador central) utilizados por aplicações de usuários, integrando ao *framework* CSBase [30] novas funcionalidades que permitem alocar os recursos computacionais necessários à execução das aplicações científicas de alto desempenho, oferecendo suporte ao gerenciamento e a reserva de processador, além da garantia de que serão efetuadas adaptações necessárias à eventuais variações no uso desse recurso. Essas novas funcionalidades oferecem benefícios para: (i) aglomerados (*clusters*); (ii) sistemas de computação em grade (*grid computers*); (iii) sistemas operacionais distribuídos; e (iv) sistemas de *middleware* baseados em objetos distribuídos com suporte para qualidade de serviço (QoS). Dessa forma, ao utilizar uma máquina conectada a um aglomerado, o usuário tem ao seu dispor não apenas os recursos de uma máquina específica, mas também os recursos do sistema distribuído, em sua diversidade e capacidade computacional.

Para validar essa hipótese, foi desenvolvido um ambiente integrado, baseado em elementos projetados com motivações diferentes: o *framework* CSBase, responsável pela distribuição e gerência dos recursos, e o sistema DSRT (*Dynamic Soft Real Time*) [9] [11], ferramenta responsável por efetuar a reserva de recursos às aplicações científicas. Esse novo ambiente oferecerá ao usuário a possibilidade de reserva do processador necessário à execução da aplicação, além da garantia de que serão efetuadas eventuais adaptações, necessárias à variação no uso desse recurso.

A intenção deste trabalho não é a de propor um mecanismo que cubra todos os aspectos necessários a uma implementação de QoS, mas sim a de estabelecer a integração entre diversos mecanismos desenvolvidos com motivações específicas, oferecendo ao usuário a possibilidade de obter alguma QoS do sistema integrado.

1.2. Estrutura da dissertação

O restante desta dissertação está organizado da seguinte forma. O Capítulo 2 apresenta alguns trabalhos recentes que envolvem propostas de ambientes com reserva e gerência de recursos para aplicações. Diversos conceitos e mecanismos empregados nos ambientes descritos nesse capítulo foram incorporados ao ambiente proposto. São descritos três ambientes que endereçam a reserva de recursos para aplicações distribuídas em sistemas multimídia.

No Capítulo 3 descreve-se a infra-estrutura, a arquitetura e as funcionalidades utilizada no *framework* CSBase, antes da integração das novas funcionalidades.

No Capítulo 4 são apresentados os conceitos do sistema DSRT, que fundamentam e sobre o qual foi desenvolvido o mecanismo oferecido pelo ambiente proposto.

No Capítulo 5 é apresentado o ambiente CSBase estendido, objeto deste trabalho. Nesse capítulo será descrito a arquitetura e funcionalidades incorporadas pelo novo ambiente assim como, serão detalhadas algumas das facilidades e mecanismos, através dos quais é oferecido o suporte à reserva de processador para as aplicações.

Finalmente, o Capítulo 6 apresenta algumas conclusões sobre o trabalho descrito nesta dissertação e aponta algumas direções onde trabalhos futuros poderão ser desenvolvidos.