2
Trabalhos Relacionados

Em ambientes onde a capacidade computacional é compartilhada, mecanismos para provisão de QoS são necessários para controlar o uso de recursos pelas aplicações e, assim, assegurar metas contratuais de desempenho. Apesar de muitos trabalhos para provisão de QoS já existirem, nota-se um crescente interesse por melhorias que englobam, principalmente, a redução da sobrecarga de monitores de máquinas virtuais e a introdução de *frameworks* de controle do uso de recursos em sistema operacionais de propósito geral. Contudo, mesmo nos trabalhos mais recentes, é possível identificar lacunas que dificultam o uso dos mecanismos propostos em determinados cenários.

Este capítulo tem por objetivo identificar as vantagens e limitações de alguns dos principais trabalhos na área de provisão de QoS e entender como esses trabalhos se relacionam com a nossa proposta de tese. Esse entendimento é de extrema importância para esclarecer por que novas iniciativas para a provisão de qualidade de serviço são necessárias, quais funcionalidades devem ser providas por essas iniciativas e de que forma.

2.1
Provisão de QoS

Muitas funcionalidades podem ser utilizadas na busca de qualidade de serviço por aplicações alocadas em um mesmo computador. Entre elas, está a reserva de recursos computacionais, que, por assegurar o acesso aos recursos a uma determinada frequência, ajuda na obtenção da qualidade de serviço esperada. Diferentes técnicas têm sido propostas para a implementação de mecanismos de reserva de recursos. Entre elas estão a alteração do núcleo do sistema operacional, a inclusão de camadas de virtualização e o desenvolvimento de ferramentas no nível do usuário. Essas técnicas diferem não só na maneira de implementar as reservas de recursos e nas consequentes complexidades de implementação e sobrecarga de trabalho gerada, mas também na rigidez imposta às reservas e no grau de adaptabilidade das regras de gerenciamento.

Recentemente, o interesse no uso compartilhado de recursos causou a redescoberta da técnica de virtualização como uma forma de isolar o desem-

penho de aplicações executadas em uma mesma máquina física [28, 6, 9, 10, 7]. Apesar de algumas melhorias nas técnicas de virtualização reduzirem substancialmente a sobrecarga de uso de recursos, tamanha melhoria não ocorreu na garantia de desempenho que diferentes técnicas de virtualização apresentam. Tipicamente, monitores de máquinas virtuais sabem somente como distribuir a capacidade de processamento entre os domínios (máquinas virtuais em execução). Eles não controlam quando e nem como esses domínios acessam o disco ou a rede de comunicação [2, 29, 11]. Assim, a virtualização ainda não é capaz de garantir o isolamento de desempenho a aplicações com uso intensivo de E/S de dados. A abordagem utilizada pela *Amazon Elastic Compute Cloud* (*Amazon EC2*), por exemplo, ilustra bem essa lacuna no controle de entrada/saída de dados: enquanto a garantia de uso de processamento no EC2 é configurável pelo tipo de instância de máquina virtual (MV) utilizada, o uso de largura de banda de disco por um domínio é proporcional à largura de processamento dedicada a esse domínio. Essa largura de processamento, por sua vez, é inversamente proporcional ao número de MVs presentes em cada servidor físico [30]. Dessa forma, para garantir uma certa largura de banda de disco, muitas vezes, é preciso alocar uma quantidade extra de processamento, mesmo que esse recurso não seja necessário para assegurar a qualidade de serviço requisitada pela aplicação.

Xen é um dos monitores de máquinas virtuais mais relevantes da literatura, sendo capaz de carregar diferentes escalonadores durante a fase de inicialização do domínio hospedeiro como, por exemplo, o EDF (*Earliest Deadline First*) o qual força domínios virtuais a executar somente durante uma fatia de tempo a cada período [31, 32]. Outros exemplos de monitores incluem o VMWare, o Solaris Containers e o OpenVZ [33, 34, 35]. No entanto, apesar de todos oferecerem a opção de atribuir reservas de processamento às máquinas virtuais, isolar o desempenho de aplicações executando dentro de cada domínio continua sendo uma tarefa do sistema operacional hospedado na MV. Assim, se uma máquina virtual utiliza um GPOS como sistema operacional, não há garantia de desempenho entre as aplicações desse domínio.

O problema de isolamento de desempenho pode ser resolvido atribuindo-se um domínio virtual a cada aplicação, mas dependendo do tipo de virtualização utilizada, essa prática pode ser computacionalmente muito custosa. Assim, em máquinas com muitas aplicações é desejável que existam mecanismos de reserva com granularidade mais fina do que as apresentadas pelas máquinas virtuais. Mecanismos assim são encontrados, por exemplo, em ferramentas de reserva no nível do usuário.

Parte da limitação na provisão de QoS encontrada na virtualização

é causada pela ausência, em sistemas operacionais de propósito geral, de um suporte para contabilidade do número de *bytes* recebidos e enviados por cada processo a dispositivos de E/S. Outro fator limitante consiste na falta de um mecanismo que permita a priorização de operações de E/S de dados independentemente do escalonador de E/S ativo no sistema. Por esse motivo, frequentemente, os núcleos de GPOSes são modificados para proverem abstrações de reserva de recursos. Esses núcleos, denominados *Resource Kernel*, garantem o acesso a determinadas partes dos recursos de uma máquina enquanto uma aplicação é executada [36, 4, 37, 38, 39]. Linux/RK e RT-Mach são dois dos principais *Resource Kernels* encontrados na literatura [40, 41]. Mais recentemente, o Xenomai, um emulador de sistemas operacionais de tempo real, também tem se destacado na área acadêmica [42].

Todos os projetos com extensões de GPOSes supracitados proveem meios para a especificação dos períodos de processamento de uma aplicação. Porém eles precisam que alterações no código do núcleo do SO sejam realizadas para tornar as reservas efetivas. Essas alterações, muitas vezes, são bastante complexas e podem inserir erros no funcionamento do sistema, além de inviabilizar a flexibilização das políticas de escalonamento ativas no SO. Exceções à limitação dessa flexibilidade são encontradas no RED-Linux [43] e no QoSOS [44]. O RED-Linux consiste em um *framework* de escalonamento que permite a alteração das políticas de escalonamento sem que modificações nos mecanismos que garantem o compartilhamento do processamento sejam necessárias. As políticas do RED-Linux são implementadas por um processo que, executando no espaço do usuário, monitora e coleta informações a respeito do montante de recursos necessário para a execução de uma determinada aplicação. Com base no montante estimado, o processo no nível do usuário calcula os parâmetros de QoS a serem garantidos e, utilizando uma API específica, os envia ao processo despachante. Esse processo despachante, por sua vez, reside no núcleo do SO e é responsável por implementar os mecanismos de garantia de qualidade de serviço. O QoSOS é um *framework* para a provisão de QoS em GPOSes. Desenvolvido para atender aplicações multimídia, o QoSOS provê flexibilidade no controle de admissão e no escalonamento de processos.

Pelas dificuldades encontradas para se modificar as funcionalidades de um SO, alguns pesquisadores da área de Sistemas Operacionais defendem a prática da total separação entre os mecanismos que implementam funcionalidades, e as políticas que determinam como as funcionalidades providas pelos mecanismos são utilizadas [45, 17, 16]. Pare esses autores, o paradigma de sistemas operacionais de micronúcleo (do termo inglês *microkernel*), quando

comparado ao paradigma de núcleo monolítico, é mais adequado a adaptações e menos propenso a erros. Isso porque no paradigma de micronúcleo busca-se reduzir o tamanho do núcleo do sistema ao prover um número mínimo de mecanismos que, por evitarem a imposição de regras, maximizam a flexibilidade de implementação do restante do sistema.

Entre os diversos SOs com micronúcleo existentes na literatura [46, 47, 48, 49, 50], há de se destacar o Hydra [17], um dos primeiros sistemas operacionais a implementar a separação dos conceitos de mecanismo e política. O núcleo do Hydra é composto quase que inteiramente de mecanismos guiados por políticas as quais, no nível do usuário, ditam regras para a utilização de recursos tais como ciclos de processamento e memória. Os parâmetros de escalonamento, determinados pelo núcleo do SO, incluem informações que ajudam na provisão de qualidade de serviço como, por exemplo, prioridades, máscara de *bits* para o uso de processadores e fatias de tempo.

Juntamente com a adoção de escalonadores justos — CFQ (*Completely Fair Queueing*) para E/S e CFS (*Completely Fair Scheduler*) para processamento — em sua versão 2.6.24, a Linux Kernel Organization[1] introduziu no núcleo de seu SO o *Task Control Groups* (TCG ou CGroup), um *framework* capaz de atribuir comportamentos arbitrários a determinados grupos de processos com o objetivo de controlar a execução dos mesmos. A ideia de subgrupos implementada pelo CGroups é semelhante às subárvores propostas pelo escalonador hierárquico de Goyal *et al.* [51]. De acordo com a documentação do *kernel* Linux, a intenção do CGroup é que subsistemas sejam acoplados ao *framework* para proverem novas funcionalidades tais como a contabilidade e a limitação de consumo de recursos que um grupo de processos pode utilizar. Até a versão de núcleo Linux mais recente (2.6.33), o CGroup é capaz de gerenciar de maneira justa os recursos de processamento e disco, e de maneira quantitativa o montante de memória a ser utilizada por cada grupo de aplicações. De acordo com a documentação do *framework*, somente operações síncronas de acesso ao disco são suportadas. Com isso, as operações de escrita armazenadas em *buffers* ainda não recebem tratamento diferenciado por grupo.

A inserção do CGroup no Linux é de grande importância para uma maior provisão de qualidade de serviço nesse sistema e expressa a atual preocupação da indústria de *software* na melhoria do uso de recursos computacionais. No entanto, poucos avanços foram feitos a fim de se prover flexibilidade e extensibilidade às políticas de escalonamento dos grupos de aplicações desse *framework*. Por esse motivo, a inserção de novas políticas continua sendo uma funcionalidade a ser implementada no código do núcleo do SO.

[1] http://www.kernel.org/

Outras iniciativas de gerenciamento de recursos propuseram a implementação de ferramentas no nível do usuário para assegurar reservas de recursos [18, 19, 20, 21, 22]. Os trabalhos descritos por Chu e Nahrstedt [18] e por Newhouse e Pasquale [20] descrevem duas implementações de ferramentas de reserva de processamento que funcionam utilizando temporizadores os quais, ao expirarem, alteram as prioridades das aplicações sendo executadas em um servidor. A primeira ferramenta, denominada DSRT (*Dynamic Soft Real Time Scheduler*), foi projetada para responder às necessidades de aplicações multimídias. Assim, ela permite classificar aplicações de acordo com os seus respectivos perfis de consumo de processamento. O DSRT provê também uma API de reserva que pode ser adicionada ao código de aplicações com o objetivo de permitir um controle mais preciso do uso de recursos.

Apesar de ser a precursora das ferramentas de reserva de recursos no nível do usuário, sendo parte de projetos de grande relevância na área acadêmica, o DSRT teve o seu desenvolvimento descontinuado. Um estudo a respeito das funcionalidades do DSRT mostrou que novos atributos deveriam ser introduzidos na ferramenta a fim de torná-la mais adequada às necessidades de alguns ambientes computacionais atuais. Seria preciso, por exemplo, alterar a forma como os processos clientes e o servidor da ferramenta se comunicam, expandir reservas na presença de processamento ocioso no sistema e prover mecanismos para alocar apenas um subconjunto de unidades de processamento em máquinas com diversos processadores. No entanto, uma análise da ferramenta revelou um código bastante complexo e difícil de ser tratado.

A *sandbox* proposta por Chang *et al.* monitora o consumo das aplicações para, em seguida, de acordo com o consumo observado, direcioná-las a um comportamento desejado [19]. Para isso, a *sandbox* utiliza um conjunto de mecanismos disponibilizados pela grande maioria dos sistemas operacionais modernos, tais como temporizadores de granularidade fina, infraestruturas de monitoração, modos de depuração, escalonamento baseado em prioridades e proteção de memória. A *sandbox* é implementada para plataformas Windows e impõe restrições quantitativas aos recursos de processamento, memória e largura de banda de rede. Uma desvantagem do trabalho de Chang é que a *sandbox* proposta por ele utiliza o método da interceptação de funções para realizar o controle de uso de recursos, uma técnica que, como será descrito na Seção 4.6, gera uma alta sobrecarga de processamento.

Uma alternativa à interceptação de funções é o uso da técnica de interposição de operações. Essa técnica, menos custosa em termos de processamento, é utilizada por Yamada e Kono [22] e por Eriksen [21] para sobrescrever operações de E/S da `libC` e, assim, controlar o acesso ao disco e à rede, res-

pectivamente. No entanto, algumas limitações podem ser encontradas nesses trabalhos. Exemplos consistem na impossibilidade de garantir acessos aos recursos, visto que as requisições de consumo de recursos são somente limitadas, na restrição de apenas uma taxa de acesso a todos os processos gerenciados e na ausência de flexibilidade de adaptação das taxas de acesso.

Ainda no nível do usuário, o trabalho proposto por Wachs *et al.* apresenta o Argon, um servidor de armazenamento o qual, implementado no topo de um sistema especializado de arquivos de rede (*cluster based storage system*), intercepta as requisições de E/S e as molda de acordo com os objetivos de desempenho das aplicações. Para isso, o Argon reimplementa o sistema de *cache* de arquivos e redimensiona dinamicamente o tamanho dos *buffers* de leitura/escrita em disco [52].

2.2 Considerações Finais

O estudo dos trabalhos descritos neste capítulo mostrou que a maioria dos sistemas que proveem reserva de recursos realiza essa tarefa por meio do uso de extensões de sistemas operacionais e de máquinas virtuais. No entanto, é sabido que alterações no código de sistemas operacionais são muito propensas a causar erros, além de serem pouco portáveis. O uso de virtualização, por outro lado, ainda gera alguns problemas decorrentes da sobrecarga de trabalho para gerenciar as máquinas virtuais, além de não isolar por completo o desempenho das máquinas virtuais [13, 15, 14].

Nos últimos anos, os avanços dados pela computação na direção do uso compartilhado de recursos fez o interesse pela área de provisão de QoS ressurgir. Em especial, nota-se um crescente interesse pela melhoria do desempenho de técnicas de virtualização, muitas vezes, apoiada na implementação de novas funcionalidades no núcleo de sistemas operacionais de propósito geral. Esse é o caso, por exemplo, do novo *framework* de gerenciamento de recursos introduzido no Linux, o CGroup. Com o CGroup, novos mecanismos de controle de consumo de recursos de processamento, memória e disco são exportados para o espaço do usuário, podendo, inclusive, ser utilizados por monitores de máquinas virtuais.

O estudo do *framework* CGroup poderia servir de base para o estudo de provisão de reserva de recursos, mas, além de ter sido incorporado ao Linux há pouco tempo, acreditamos que uma ferramenta totalmente desenvolvida no nível do usuário, tal como alguns trabalhos relacionados descritos, traria mais contribuições ao nosso estudo. Com a ferramenta, gostaríamos de investigar a possibilidade de uma implementação facilmente portável para outros sistemas

operacionais de propósito geral e que nos permitisse verificar a viabilidade de se implementar uma ferramenta que provesse flexibilidade no desenvolvimento, na configuração e na combinação das políticas de escalonamento. Gostaríamos também que essa ferramenta provesse reserva de recursos de granularidade fina, uma funcionalidade que, assim como a flexibilidade no uso das políticas de escalonamento, não é oferecida pelo *framework* CGroup e nem pelas diversas técnicas de virtualização.

A partir da análise dos trabalhos relacionados à provisão de QoS no nível do usuário, concluímos que nenhuma das ferramentas de reserva de recursos estudadas apresentava todas as características que gostaríamos de encontrar para a realização do estudo. Problemas foram encontrados, por exemplo, nos tipos de recursos controlados pelas ferramentas, pois apesar da nossa investigação ter como alvo a limitação e garantia de reservas de processamento e largura de banda de disco, nenhuma ferramenta contemplava o gerenciamento desses recursos conjuntamente. Assim, decidimos por implementar uma nova ferramenta de reservas.

Diferentemente da ferramenta de reserva de recursos no nível do usuário de maior relevância na literatura, o DSRT, o qual foi proposto para atender as necessidades de aplicações multimídia, a nossa ferramenta foi projetada para atender às necessidades de aplicações de computação intensiva em cenários como, por exemplo, grades oportunistas, sistemas de *utility computing* e arquiteturas multiprocessadas. Consequentemente, algumas funcionalidades da nossa ferramenta consistem na capacidade de expandir taxas de reserva na presença de processamento ocioso no sistema e na provisão de mecanismos para alocar apenas um subconjunto de unidades de processamento em máquinas com diversos processadores.

No desenvolvimento deste trabalho, as reservas de recursos foram utilizadas com o objetivo de garantir desempenho às aplicações somente. Porém, muitas outras utilidades podem ser atribuídas à ferramenta como, por exemplo, o controle do nível de energia de dispositivos móveis, a redução do aquecimento produzido por computadores, a construção de ambientes de testes com variadas capacidades computacionais (testes de escalabilidade) e o estudo de diferentes políticas de escalonamento. Ao considerar essas diversas áreas, muitos outros trabalhos relacionados à nossa ferramenta podem ser encontrados [53, 54, 55].

Por tratar apenas da parte de provisão de QoS, a nossa ferramenta não provê um mecanismo de escalonamento orientado às necessidades da aplicação, tal como o RED-Linux faz, ainda que essa tarefa possa ser realizada por mecanismos externos à nossa ferramenta. Para isso seria preciso a integração da ferramenta com uma arquitetura de gerenciamento de recursos. Arquiteturas

desse tipo devem estar presentes em infraestruturas computacionais compartilhadas, coordenando a execução de aplicações desde a submissão das mesmas até a sua finalização.

Apesar de uma arquitetura de gerenciamento não ser tratada no nosso estudo, entendemos a importância de se compreender a relação das ferramentas de reserva com esse cenário. Assim, no próximo capítulo, a ferramenta de apoio desenvolvida é brevemente contextualizada em uma arquitetura de gerenciamento de recursos e tem seus detalhes de projeto e implementação apresentados.