Escalonamento orientado por qualidade de serviço em infraestruturas de computação na nuvem

Giovanni Farias da Silva[1,2], **Francisco Brasileiro**[1], **Raquel Lopes**[1]

[1]Universidade Federal de Campina Grande (UFCG)
Programa de Pós-Graduação em Ciência da Computação
Campina Grande, PB, Brasil

[2]Universidade Federal Rural de Pernambuco (UFRPE)
Belo Jardim, PE, Brasil

giovanni.farias@ufrpe.br, {fubica,raquel}@computacao.ufcg.edu.br

Abstract. *Cloud computing providers offer multiple service classes to deal with workload heterogeneity. Classes are distinguished by their expected Quality of Service (QoS), which is defined in terms of Service Level Objectives (SLO). A priority-based scheduling policy is commonly used to guarantee that requests submitted to the different service classes achieve the desired QoS. However, the QoS delivered during resource contention periods may be unfair to certain users. We proposed QoS-driven scheduling policies which take the SLOs and actual QoS delivered to each request into account when making decisions. We used simulation experiments fed with traces from a production system to compare the QoS-driven policy with a priority-based one, and show the benefits of QoS-driven scheduling.*

Resumo. *Provedores de computação na nuvem oferecem múltiplas classes de serviço para lidar com a heterogeneidade da carga de trabalho. Essas classes são diferenciadas pela Qualidade de Serviço (QoS) esperada, que é definida em termos de Objetivos de Níveis de Serviço (SLO). Uma política de escalonamento baseada em prioridades é uma forma comum de permitir que requisições de diferentes classes alcancem a QoS desejada. No entanto, em períodos de contenção de recursos, a QoS oferecida pode ser injusta para certos usuários. Este trabalho propõe o escalonamento orientado por QoS, que usa uma política baseada no SLO e na QoS de cada requisição em um dado momento. Experimentos de simulação usando dados de um sistema real em produção foram executados para comparar as diferentes políticas e mostrar os benefícios do escalonador proposto.*

1. Introdução

Um dos principais desafios para grandes provedores de Computação na Nuvem é lidar com infraestruturas de computação bastante complexas e de larga escala, sujeitas a cargas de trabalho heterogêneas que variam ao longo do tempo [Verma et al. 2014]. Essas características implicam que a gestão da infraestrutura deve lidar com diferentes requisitos do usuário e ampla variabilidade na demanda de recursos, o que geralmente leva à subutilização da infraestrutura e aumento do custo operacional.

Provedores de nuvem tradicionais oferecem pelo menos uma classe de serviço com garantias de Qualidade de Serviço (QoS, do inglês *Quality of Service*). A QoS prometida para cada classe é definida através de metas específicas. Essas metas são estabelecidas pelos "Objetivos de Nível de Serviço" (SLO, do inglês *Service Level Objectives*) encontrado no "Contrato de Nível de Serviço" (SLA, do inglês *Service Level Agreement*) acordado entre o cliente e o provedor. O SLA também define as penalidades que são aplicadas aos provedores quando os SLOs não são atendidos.

Com o objetivo de aumentar a utilização dos recursos (consequentemente, a lucratividade), uma abordagem comum dos provedores é oferecer múltiplas classes de serviço, cada uma com uma QoS prometida. Em particular, grandes provedores oferecem sua capacidade temporariamente ociosa de forma oportunista, essencialmente sem qualquer garantia de QoS [Marshall et al. 2011]. Entretanto, a ausência de uma expectativa mínima de QoS restringe as aplicações que podem se beneficiar desses recursos; além disso, os recursos oferecidos oportunisticamente geralmente são vendidos a preços bem menores do que os normais (por exemplo, os serviços Amazon EC2 spot e Google Cloud preemptible). Carvalho *et al.* [Carvalho et al. 2014, Carvalho et al. 2015] mostraram que um provedor pode agregar mais valor aos seus recursos ociosos, oferecendo-os por meio de uma ou mais classes de serviço com QoS bem definida. Cada nova classe é associada a um SLO de longo prazo. Por causa da QoS prometida, o provedor consegue cobrar mais pelos recursos alocados a requisições dessas classes, em comparação àquelas oferecidas oportunistamente. Isso permite que o provedor aumente a utilização dos recursos de forma mais rentável [Xu and Zhu 2015]. Essas novas classes são úteis para aplicações que podem aceitar serviço ligeiramente degradado, mas ainda precisa de garantias de QoS moderadas (por exemplo, pipelines não interativos, como indexação da web e transcodificação de vídeo) [Cirne and Frachtenberg 2012]. A fim de atender uma variedade de requisitos do usuário e melhorar a utilização da infraestrutura, consideramos os provedores que oferecem várias classes de serviço, cada uma com um esquema diferente de preço e QoS.

O gerenciamento eficiente de seus recursos permite que o provedor possa cumprir os SLAs das diferentes classes de serviço, ao mesmo tempo que reduz os custos envolvidos com o provisionamento do serviço. As atividades de gerenciamento podem ser divididas em três etapas principais: *Planejamento de Capacidade*, *Controle de Admissão* e *Escalonamento* [Carvalho et al. 2015]. O *Planejamento de Capacidade* define a quantidade de recursos adequada para que o provedor consiga atender a demanda esperada. Esta etapa estabelece a capacidade da infraestrutura do provedor para um período relativamente longo (horizonte de meses). Na etapa de *Controle de Admissão* são estabelecidos critérios de admissão/rejeição de novas requisições. A possível rejeição de parte das requisições tem o intuito de não permitir que novas requisições gerem uma carga que possa afetar negativamente — a ponto de causar violações de SLOs — o desempenho daquelas que já foram admitidas anteriormente. Por último, na fase de *Escalonamento*, decide-se quais requisições terão recursos alocados em um dado instante e quais os servidores proverão os recursos para as mesmas. Em momentos em que há contenção de recursos (*i.e.* quando não é possível alocar recursos para todas as requisições ativas no sistema), um subconjunto das requisições permanecem pendentes, esperando que recursos ocupados sejam liberados. O escalonador é responsável por escolher quais das requisições devem estar em execução e quais devem ficar pendentes (se necessário). Quando não há contenção por recursos, qualquer escalonador consegue alocar todas as requisições ativas. Portanto,

é durante os momentos onde há contenção por recursos que as diferentes políticas de escalonamento podem fazer a diferença.

O foco de nosso trabalho *é a etapa de escalonamento*. Os usuários submetem requisições para as diferentes classes de serviço disponíveis. As requisições especificam os recursos necessários (por exemplo, CPU, RAM, etc.), que são agrupados usando abstrações de isolamento, como máquinas virtuais ou contêineres. Neste trabalho, chamamos esse pacote de uma instância de alocação, ou simplesmente uma instância. Uma vez que uma requisição é admitida, o escalonador trabalha para alocar uma ou mais instâncias para atender à requisição, considerando os SLOs prometidos no SLA associado à requisição da classe de serviço.

Diferentes escalonadores executam políticas de escalonamento distintas, que buscam atender objetivos diversos. No contexto de múltiplas classes de serviço, diversos escalonadores de provedores de larga escala usam políticas *baseadas em prioridade* [Boutin et al. 2014, Burns et al. 2016, Curino et al. 2014, Delimitrou et al. 2015, Dubey and Agrawal 2013, Karanasos et al. 2015, Schwarzkopf et al. 2013, Vavilapalli et al. 2013, Verma et al. 2015]. Esses escalonadores diferenciam as classes de serviço através da associação de uma prioridade para cada uma delas. Quanto maior é a QoS prometida para a classe, maior será a prioridade atribuída a ela. Nesse cenário, cada instância é relacionada com a mesma prioridade da classe de serviço da requisição. A prioridade de uma instância, portanto, funciona como um *proxy* para a QoS que lhe é prometida. Nesse sentido, um escalonador baseado em prioridade pode preemptar os recursos alocados para uma instância em benefício de uma outra, desde que a última tenha uma prioridade maior que a primeira. Entre requisições de mesma prioridade, usualmente a mais antiga tem preferência no momento da alocação [Burns et al. 2016]. Portanto, nesse contexto, as requisições com prioridades mais altas têm menos chances de serem preemptadas, e, consequentemente, elas têm mais chances de alcançar uma melhor QoS que as requisições com prioridades mais baixas.

No entanto, apesar de considerar a QoS prometida para a classe de serviço, a política de escalonamento baseada em prioridade pode tomar decisões ineficientes em pelo menos duas situações, quando há contenção por recursos: (i) as requisições com prioridades mais baixas têm maiores chances de não receber a QoS esperada. Isso se deve à possibilidade delas sempre serem preemptadas em benefício de requisições com prioridades mais altas. Isso ocorre inclusive quando as últimas já estiverem recebendo QoS acima da esperada; e (ii) dado que uma requisição não é preemptada em benefício de uma outra de mesma classe, a QoS entregue para requisições de uma mesma classe pode apresentar alta variabilidade. Isso ocorre quando, em períodos com contenção de recursos, duas requisições de mesma classe estão competindo pelo mesmo recurso.

Em resumo, o escalonador baseado em prioridade não foi concebido para respeitar a QoS prometida para todas as classes de serviço simultaneamente. Enquanto houver requisições com prioridades mais altas necessitando de recursos, o escalonador alocará qualquer recurso livre para elas. Além disso, caso seja necessário, ele sempre preemptará recursos de requisições com prioridades mais baixas para liberar recursos para outras de maior prioridade, independente das QoSs atuais das requisições envolvidas. Por fim, uma vez que os recursos são alocados para uma requisição, eles nunca serão preemptados em benefício de uma outra de mesma classe. Por esta razão, em períodos com contenção de

recursos, é possível que aconteça alta variabilidade na QoS entregue para requisições de uma mesma classe (*i.e.* o provisionamento de recursos não é justo nesse cenário).

Uma possível solução para essas situações seria fazer com que o provedor sempre trabalhasse com uma infraestrutura super provisionada. No entanto, os custos envolvidos em manter essa infraestrutura proíbem esta alternativa. É importante destacar que as atividades de planejamento de capacidade e controle de admissão, que operam em conjunto com o escalonamento, sempre buscam evitar cenários de super provisionamento, mantendo os custos operacionais o mais baixo possível, ao mesmo tempo que tentam evitar também o sub provisionamento, para permitir que a QoS entregue às instâncias fique em níveis aceitáveis. Nesse contexto, pode-se inferir que o provedor tem como objetivo operar em cenários com contenção moderada, onde a infraestrutura e carga de trabalho são dimensionadas de forma a manter alta utilização de recursos e ainda entregar a QoS prometida para todas as requisições. No entanto, imprecisões tanto no controle de admissão como no planejamento de capacidade podem causar períodos com alta contenção, por isso, o escalonador precisa lidar com essas situações.

2. Objetivos

O objetivo geral deste trabalho foi propor e avaliar uma nova política de escalonamento que pudesse *lidar com a QoS prometida para todas as classes de serviço simultaneamente e tratar de forma mais justa requisições de uma mesma classe*, particularmente em períodos com contenção de recursos. Por tratamento justo, entende-se que as requisições de uma mesma classe devem receber QoS semelhante quando estiverem competindo pelos mesmos recursos, inclusive quando não for possível entregar a QoS prometida para todas as requisições. Tendo em vista este objetivo principal, os seguintes objetivos específicos foram definidos:

1. Propor uma política de escalonamento que tome decisões com base na QoS que estiver sendo entregue para cada requisição no sistema no momento da tomada de decisão, bem como em suas respectivas metas de QoS;
2. Identificar como os rastros de execução de provedores públicos disponíveis podem ser utilizados para avaliar esta nova política de escalonamento;
3. Avaliar a política proposta através da comparação dos resultados obtidos por ela com os obtidos por escalonadores baseados em prioridades;
4. Analisar a viabilidade da nova política com base em aspectos práticos relacionados, por exemplo, ao desempenho do escalonador e a forma como tipicamente os usuários do provedor submetem suas cargas de trabalho.

3. Escalonamento Orientado por QoS

Esta tese propõe uma nova política de escalonamento para provedores de computação na nuvem, denominada **Escalonamento Orientado por QoS**. Neste cenário, o escalonador toma decisões levando em conta a QoS entregue para cada uma das requisições no sistema no momento da tomada de decisão e suas correspondentes metas de QoS. Esta política tem basicamente dois objetivos: (i) atender as metas de QoS das requisições admitidas, quaisquer que sejam suas classes de serviço; e, (ii) oferecer tratamento justo para requisições de uma mesma classe, entregando QoS semelhante para as requisições que competem pelo mesmo recurso, particularmente em períodos com contenção de recursos. Esses objetivos são alcançados através de um novo mecanismo de preempção. Este

novo mecanismo preempta requisições cuja QoS esteja excedendo a QoS esperada, e, usa os recursos liberados para alocar recursos para requisições cuja QoS esteja abaixo de sua respectiva meta (ou mais próxima de violá-la). Na prática, o Escalonador Orientado por QoS está sempre trabalhando para diminuir a diferença entre a QoS entregue para as requisições e suas respectivas metas de QoS.

A relevância deste trabalho pode ser destacada tanto do ponto de vista do provedor quanto de seus consumidores. Na perspectiva do provedor, este pode usar seus recursos de forma mais efetiva em períodos em que não há recursos ociosos. A política proposta visa alocar os recursos de forma que uma maior quantidade de requisições tenha seus SLAs cumpridos, independentemente de suas respectivas classes. Já na perspectiva do usuário, este sempre vai receber a melhor QoS que o provedor pode entregar para a classe solicitada, com base na demanda atual do provedor e no provisionamento justo (igualitário) entre requisições de mesma classe.

4. Resultados

Nós comparamos o desempenho da política orientada por QoS com o de uma do estado-da-prática baseada em prioridades. Nossa comparação usa modelos de simulação para as políticas de escalonamento, os quais foram validados por meio de experimentos de medição. Na validação dos modelos de simulação, utilizamos o popular sistema Kubernetes [Burns et al. 2016] — que já incorpora um escalonador padrão que segue uma política baseada em prioridades — e uma implementação de prova de conceito do escalonador orientado por QoS para o Kubernetes. As simulações foram alimentadas com amostras obtidas a partir de um rastro de execução real de um *cluster* em produção da Google[1].

Nossos resultados mostraram que quando não há contenção de recursos ou o nível de contenção é muito baixo, ambos os escalonadores são capazes de fornecer a QoS prometida e se comportam essencialmente da mesma maneira. Por outro lado, quando o nível de contenção é muito alto, o escalonador orientado por QoS fornece QoS que geralmente está mais próxima da meta estabelecida do que aquela fornecida pelo escalonador baseado em prioridade, sem diferenças significativas na porcentagem de requisições cujos SLOs são violados. Finalmente, quando o nível de contenção é moderado, o escalonador orientado por QoS aumenta substancialmente a QoS entregue às requisições da classe que promete a QoS mais baixa, sem afetar o das outras classes.

Além disso, com base na prática atual de grandes provedores públicos, os resultados mostraram que as penalidades incorridas pelo uso do escalonador baseado em prioridade podem ser, em média, até aproximadamente 2 vezes mais altas que aquelas incorridas pelo escalonador orientado por QoS. A Figura 1 mostra que nos cenários com menor contenção de recursos (infraestruturas com tamanhos N e $0,9N$), o comportamento do escalonador orientado por QoS (produzindo déficits de QoS menores e menos variáveis) impactou substancialmente no custo total das penalidades do provedor. Já no caso com infraestrutura menor (*i.e.* maior contenção de recursos), a contenção durante períodos críticos foi bastante alta, deixando pouca margem de manobra para operação do escalonador orientado por QoS. Por isso, o custo com as penalidades resultante do uso do escalonador baseado em prioridade não foi tão mais alto quanto nos cenários anteriores.

[1]O rastro de execução da Google está disponível para download em `https://github.com/google/cluster-data/blob/master/ClusterData2011\_2.md`.

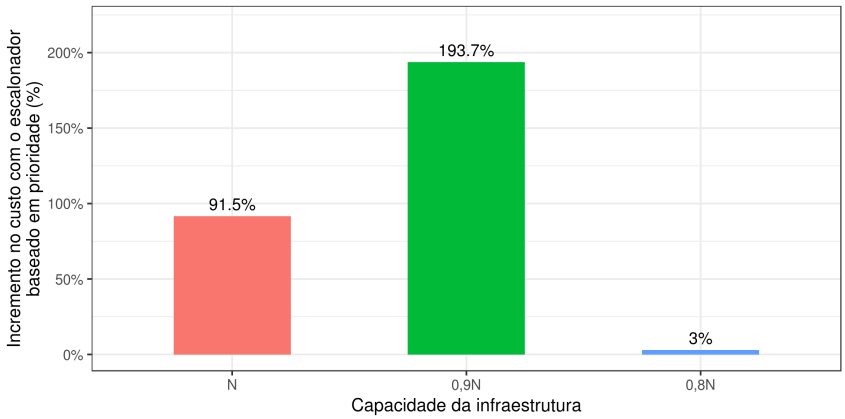

Figura 1. Incremento nos custos com penalidades quando o escalonador baseado em prioridade é usado em comparação com o orientado por QoS.

Além disso, analisamos o custo operacional do escalonador orientado por QoS. Identificamos que, sob o aspecto do processamento necessário para a tomada de decisão, a sobrecarga foi maior quando o escalonador orientado por QoS foi executado. Isso ocorre devido a necessidade de calcular as métricas utilizadas sempre que uma decisão precisar ser tomada. No entanto, ainda há bastante espaço para a implementação de otimizações, tal como um mecanismo de *cache* que já é implementado no escalonamento baseado em prioridade.

Por fim, modelamos quatro abordagens distintas para computar a QoS (disponibilidade) fornecida para aplicações que necessitam de mais de uma instância. No caso do Kubernetes, uma única requisição pode estar associada a múltiplas instâncias (pods, no jargão do Kubernetes). Isso muda a forma como a QoS de uma requisição é definida. Nós definimos quatro semânticas para a QoS de uma requisição nomeadas de independente, concorrente, média e agregada. Nesse contexto, um provedor pode suportar uma ou mais dessas semânticas, e, o usuário pode escolher a semântica de seu interesse no momento da submissão de sua requisição.

5. Contribuições

A **política de Escalonamento Orientada por QoS** (descrita no capítulo 3 da tese) é a principal contribuição da tese. Após extensa pesquisa bibliográfica, identificamos limitações no escalonamento baseado em prioridades utilizado no estado-da-prática por provedores de nuvem. Na etapa de concepção das ideias da política de escalonamento orientado por QoS, juntamente com nosso parceiro industrial (Ericsson Telecomunicações), depositamos a **patente** "*Method and resource manager for scheduling of instances in a data centre*" com número de depósito PCT/SE2017/050375. Atualmente, a patente está na etapa de *disclosure*, já tendo superado as etapas de depósito da documentação e exame técnico [da Silva et al. 2017].

Também realizamos a **avaliação experimental** (capítulos 4, 5 e 6 da tese) e a **análise de viabilidade** (capítulos 5 e 7 da tese) do escalonamento orientado por QoS. Essas avaliações foram executadas em etapas incrementais. A nova política de escalonamento foi apresentada no **artigo** "*Escalonamento justo em infraestruturas de nuvem com múltiplas classes de serviço*", publicado no Simpósio Brasileiro de Redes de Computado-

res e Sistemas Distribuídos (SBRC) 2019 [da Silva et al. 2019]. Neste artigo comparamos as disponibilidades entregues pelo escalonamento orientado por QoS com as obtidas com escalonamento baseado em prioridade. Essa comparação se deu através de modelos de simulação. Os resultados mostraram que a nova política foi capaz de entregar QoS satisfazendo os SLOs de todas as requisições em uma parcela significantemente maior dos cenários. Este artigo foi *premiado como o melhor artigo* da trilha principal do evento.

Em seguida, aperfeiçoamos a nova política através da introdução de mecanismos para limitação de preempções e de funções configuráveis a serem utilizadas na escolha de servidores para alocação e de instâncias a serem preemptadas (quando necessário). Esse incremento, juntamente com a eliminação de limitações da avaliação inicial (tais como a homogeneidade da infraestrutura e a falta de *placement constraints*) e avaliações sob novos aspectos (tais como custo com penalidades decorrentes de violações de SLOs e justiça no provisionamento de recursos), resultaram na publicação do **artigo** "*QoS-driven scheduling in the cloud*" [da Silva et al. 2020] no periódico *Journal of Internet Services and Applications*.

Além disso, a análise de viabilidade se deu com **a implementação do escalonador** para o sistema Kubernetes[2] (padrão *de facto* em gerência de contêineres em ambientes de computação na nuvem). Este foi usado para validar os modelos de simulação utilizados e serve como prova de conceito para a nova política de escalonamento.

Por fim, após a defesa da tese, o Kubernetes passou a prover uma arquitetura plugável para o escalonador[3]. Essa arquitetura tem como objetivo tornar a política de escalonamento customizável, facilitando a integração de novos escalonadores e a adoção desses escalonadores em sistemas de produção. Nesse contexto, o escalonador possui um conjunto de pontos de extensões, que possibilitam que as funcionalidades sejam implementadas por plugins, enquanto que o corpo principal do escalonador torna-se simples e fácil de manter. Nós desenvolvemos um novo protótipo para o escalonador orientado por QoS, implementando os plugins necessários. Os resultados preliminares indicam que a sobrecarga para a tomada de decisões do novo escalonador foi reduzida consideravelmente. Atualmente, temos investigado o impacto de diferentes semânticas para a QoS da requisição (descritas no capítulo 7), bem como o impacto do escalonamento orientado por QoS na QoS das aplicações que executam nos recursos alocados. Os resultados dessa avaliação serão reportados no artigo "*Availability-driven scheduling in Kubernetes*", atualmente em desenvolvimento para ser submetido para o periódico *IEEE Transactions on Cloud Computing*.

Referências

Boutin, E., Ekanayake, J., Lin, W., Shi, B., Zhou, J., Qian, Z., Wu, M., and Zhou, L. (2014). Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *OSDI*, volume 14, pages 285–300.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. (2016). Borg, omega, and kubernetes. *Commun. ACM*, 59(5):50–57.

[2]Informações sobre o sistema Kubernetes estão disponíveis em `https://kubernetes.io/`.

[3]`https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/`

Carvalho, M., Cirne, W., Brasileiro, F., and Wilkes, J. (2014). Long-term slos for reclaimed cloud computing resources. In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, pages 20:1–20:13, New York, NY, USA. ACM.

Carvalho, M., Menascé, D., and Brasileiro, F. (2015). Prediction-based admission control for iaas clouds with multiple service classes. In *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, pages 82–90. IEEE.

Cirne, W. and Frachtenberg, E. (2012). Web-scale job scheduling. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–15. Springer.

Curino, C., Difallah, D. E., Douglas, C., Krishnan, S., Ramakrishnan, R., and Rao, S. (2014). Reservation-based scheduling: If you're late don't blame us! In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, pages 2:1–2:14, New York, NY, USA. ACM.

da Silva, G. F., Brasileiro, F., Lopes, R., Carvalho, M., and Turull, D. (2017). Method and resource manager for scheduling of instances in a data centre.

da Silva, G. F., Brasileiro, F. V., Lopes, R. V., Morais, F. J. A., Carvalho, M., and Turull, D. (2020). Qos-driven scheduling in the cloud. *J. Internet Serv. Appl.*, 11(1):9.

da Silva, G. F., Lopes, R., Brasileiro, F., Carvalho, M., Morais, F., Mafra, J., and Turull, D. (2019). Escalonamento justo em infraestruturas de nuvem com múltiplas classes de serviço. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 636–649, Porto Alegre, RS, Brasil. SBC.

Delimitrou, C., Sanchez, D., and Kozyrakis, C. (2015). Tarcil: reconciling scheduling speed and quality in large shared clusters. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 97–110. ACM.

Dubey, S. and Agrawal, S. (2013). Qos driven task scheduling in cloud computing. *Int. J. Comput. Appl. Technol. Res*, 2(5):595–600.

Karanasos, K., Rao, S., Curino, C., Douglas, C., Chaliparambil, K., Fumarola, G. M., Heddaya, S., Ramakrishnan, R., and Sakalanaga, S. (2015). Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In *USENIX Annual Technical Conference*, pages 485–497.

Marshall, P., Keahey, K., and Freeman, T. (2011). Improving utilization of infrastructure clouds. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, pages 205–214, Washington, DC, USA. IEEE Computer Society.

Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., and Wilkes, J. (2013). Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 351–364, New York, NY, USA. ACM.

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. (2013). Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM.

Verma, A., Korupolu, M., and Wilkes, J. (2014). Evaluating job packing in warehouse-scale computing. In *2014 IEEE Int'l Conf. on Cluster Computing, CLUSTER 2014, Madrid, Spain*, pages 48–56.

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. (2015). Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 18:1–18:17, New York, NY, USA. ACM.

Xu, J. and Zhu, C. (2015). Optimal pricing and capacity planning of a new economy cloud computing service class. In *2015 International Conference on Cloud and Autonomic Computing*, pages 149–157. IEEE.