Uma análise do consumo de energia de ambientes de processamento de dados massivos em nuvem

Nestor D. O. Volpini[1,2], Vinicius S. Conceição[1], Raphael L. Pontes[1], Dorgival Guedes [1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG

{nestor,vconceicao,raphaelluciano,dorgival}@dcc.ufmg.br

[2]Centro Federal de Educação
Tecnológica de Minas Gerais (CEFET-MG) – Divinópolis, MG

nestor@div.cefetmg.br

Resumo. *As áreas de processamento de dados massivos (PDM, ou big-data) e de computação em nuvem têm se desenvolvido de forma integrada. Assim, PDM vem se consolidando como um dos maiores consumidores de recursos em datacenters que, por sua vez, são responsáveis por demandar em torno de 2% da energia global. Entender como elementos como o ambiente de virtualização e grau de paralelização dessas aplicações afetam esse consumo é, portanto, uma necessidade premente. Este artigo utiliza uma solução de monitoração abrangente para avaliar como o consumo de energia de aplicações big-data virtualizadas varia em função dos recursos alocados.*

Abstract. *Massive data processing (big-data) related fields and cloud computing have been growing conjointly. Thus, data processing is among the largest resource consumers in datacenters, consuming around 2% of global energy. Comprehension of how elements such as virtualized environments and applications' parallelization degree affect such consumption is therefore an urgent need. This article relies on a monitoring solution that provides performance metrics, data mining application logs, and data produced in distributed environments to assess how power consumption of virtualized big-data applications varies on allocated resources.*

1. Introdução

O modelo de computação em nuvem tem se tornado cada vez mais presente, já que oferece acesso compartilhado via rede a um conjunto de recursos computacionais de forma conveniente, com esforço reduzido para sua configuração, além de ser escalável. A evolução da Internet permitiu a oferta de nuvens computacionais, que possuem vantagens para quem precisa destes serviços, mas não tem condições ou interesse de fazer grandes investimentos em infraestrutura. Computação em nuvem viabiliza a escalabilidade imprevisível de um negócio sem preocupar seu dono com o provisionamento extra de recursos. Além disso, o uso de computação em nuvem estabelece um modelo de remuneração por serviços, ou seja, pague pelo que processar (*pay-per-use*).

Qualquer que seja o modelo de serviço oferecido, seja *Software as a Service* (SaaS), *Platform as a Service* (PaaS) ou *Infrastructure as a Service* (IaaS), o processamento de qualquer tarefa efetivamente acontece dentro da nuvem, que por sua vez está

fisicamente estabelecida em *datacenters*. *Datacenters* podem ser caracterizados por ocupar desde de um pequeno conjunto de poucos servidores até alguns milhares de servidores acondicionados em instalações elevadas [Koomey 2011].

O consumo de energia em *datacenters* sempre mereceu atenção. Além de ser um recurso de custo relevante, de acordo com o relatório publicado por Berkeley Lab [Shehabi et al. 2016], no ano de 2014 1,8% de toda energia consumida apenas nos Estados Unidos foi utilizada por *datacenters*. No período entre os anos de 2010 a 2014 houve um aumento de 4% na demanda. Espera-se o mesmo crescimento entre os anos de 2014 a 2020 (4%). Além disso, a matriz de produção energética se sustenta emitindo preocupantes quantidades de carbono na atmosfera de acordo com o relatório do NRDC(Natural Resources Council) [Whitney and Delforge 2014].

O processamento em *datacenters* pode ser caracterizado por dois tipos de cargas bem diferentes: podem ser utilizados para serviços *web* e para processamento de aplicações sobre grandes volumes de dados conhecidos como *big data*, também denominados dados massivos. Serviços *web* são executados predominantemente em máquinas virtuais (ou V.M.'s) [1] há um bom tempo [Zhang et al. 2010]. Já o processamento de *big data* normalmente é realizado sem virtualização (ou *bare metal*), mas já se percebe uma tendência de também se tornarem virtualizados [Hashem et al. 2015].

Independente da carga, há consideráveis vantagens em se trabalhar com sistemas virtualizados: flexibilidade, disponibilidade, escalabilidade, melhor aproveitamento do *hardware*, segurança, custo, adaptabilidade a variações de carga de trabalho, facilidade no balanceamento de carga e também na execução de versões antigas de aplicações que por algum motivo não serão reescritas mas que ainda sejam eventualmente necessárias [2] [Sahoo et al. 2010].

Muito já se estudou em economia de energia voltada para o uso de máquinas virtuais em *datacenters*. Como exemplo, podemos citar [Dupont et al. 2012] que apresenta um *framework* para alocar máquinas virtuais em um *datacenter* de modo a respeitar um conjunto de regras, entre elas economizar energia. Existe também um estudo que propõe uma arquitetura de *datacenter* eficiente energeticamente, que avalia o custo de migração de máquinas virtuais e a relação de compromisso entre consumo e desempenho [Ye et al. 2010]. No estudo feito por [Kansal et al. 2010] é apresentada uma abordagem para medir o consumo de uma V.M. em um sistema virtualizado sem o uso de instrumentos externos como wattímetros, para fins de provisionamento de carga e avaliação de custo com energia.

Portanto muito se estudou sobre as cargas tipicamente processadas em *datacenters* que lidam com máquinas virtuais, porém é importante observar um considerável crescimento no processamento de outro tipo de carga que tem se tornado comum, que lida com processamento de dados massivos [Mashayekhy et al. 2014, Gu et al. 2011]. Para processar dados massivos usualmente um conjunto de computadores físicos são ocupados de forma colaborativa, trabalhando em paralelo, para que cada um assuma uma parte da carga de processamento. Isso caracteriza uma mudança de carga, uma vez que o volume de dados a ser processado não é suportado por sistemas de servidores convencionais.

[1] Abreviatura de uso frequente, referenciando em inglês: Virtual Machines

[2] Também chamadas de "aplicações legadas"

Observa-se nos últimos anos um aumento do uso de sistemas de processamento de dados massivos, tais como Hadoop e mais recentemente Spark, além de outros. Hadoop pode ser visto como a implementação de *software* livre do paradigma de programação MapReduce, que se caracteriza por ser um modelo de processamento distribuído de grandes volumes de dados, de elevada robustez e escalabilidade. Já Spark se propõe a ser uma plataforma de processamento de dados massivos que adota um modelo de programação mais flexível. Spark se apresenta como um *framework* veloz por processar dados em memória, e de uso geral para processamento de dados massivos.

Portanto, há estudos que visam economizar energia em ambientes virtualizados, no entanto, nem sempre o escalonamento adequado de recursos para processar dados massivos é eficiente, especialmente no consumo de energia [Leverich and Kozyrakis 2010]. Ao longo deste artigo apresentaremos resultados obtidos através de experimentos, que demonstram que não é trivial definir a uma boa escolha na configuração de V.M.'s, ou mesmo sua política de distribuição dentro da estrutura, seja visando obter desempenho na conclusão da tarefa, ou para economizar energia, ao processar dados massivos.

Para isso, variadas configurações foram testadas em um ambiente de nuvem, sempre com seu consumo sistematicamente monitorado, além de outras métricas e *logs*, ao executar aplicações PDM. Para monitorá-las foi utilizado emphSeshat [Conceição et al. 2018], um sistema robusto e modularizado de monitoração desenvolvido em camadas e baseado em *softwares open-source*. Como carga para os testes, dois algoritmos de mineração de dados, Twidd e Eclat, em implementações paralelas foram testados sobre uma base de dados de 8 GB coletada a partir do Twitter. Pretendemos com esse estudo mostrar que ainda há oportunidades que consideramos significativas para economizar energia ao processar dados massivos de forma virtualizada em uma nuvem, por meio do escalonamento adequado de recursos.

2. Trabalhos relacionados à economia de energia em *datacenters*

Um maneira simples de economizar energia em *datacenters* vem da própria evolução da indústria de processadores: a incorporação de múltiplos núcleos em processadores os torna capazes de executar mais tarefas paralelas sem aumentar significativamente seu consumo [Ramanathan 2006]. Nos últimos anos é possível observar o desenvolvimento de *chips* multiprocessados de elevado desempenho e grau de paralelismo consumindo até 50 % menos energia [Kgil et al. 2006]. Também é possível atuar variando de forma combinada a frequência e a tensão de operação nos processadores do *datacenter*, técnica denominada DVFS (*Digital Voltage Frequency Scaling*), em função da carga do ambiente em que se encontram [Kim et al. 2009]. Processadores gastam menos energia ao trabalhar com um *clock* mais baixo. Mais recentemente, GPU's tem apresentado crescente utilização para processar grande volumes de dados e também são motivo de estudos do ponto de vista de seu consumo [Ferro et al. 2017].

Outra forma de economizar energia consiste em estabelecer arquiteturas de *datacenters* voltadas a um perfil energeticamente eficiente também, seja pela modificação da estrutura de rede, pela escolha dos processadores ou mesmo da forma de acesso a dados persistentes e a maneira como são replicados. Também é possível atuar nas redes de comunicação dentro do *datacenter* escolhendo sua topologia [Abts et al. 2010], ou mesmo provando que há ganhos significativos em possuir controle independente de cada

link de comunicação. Há estudos em que se opta por adotar processadores de menor capacidade, mas com baixo consumo na composição de um *datacenter*, ainda que o objetivo seja processar grandes volumes de dados. Há interessantes resultados em [Gu et al. 2011] para o processamento de uma carga MapReduce, onde está evidente que a melhor arquitetura para economizar energia com esse tipo de processamento é uma construção híbrida de servidores de pequeno porte, constituídos por processadores Atom trabalhando em conjunto com unidades de aceleração de compressão e descompressão de dados. Em um sistema denominado FAWN [Andersen et al. 2009], nodos de processamento de dados massivos são elaborados de modo que a informação persistente fica armazenada em memórias Flash e com acesso puramente log-estruturado. Os resultados obtidos mostram que FAWN é capaz de economizar até duas ordens de grandeza em consumo quando comparado com sistemas que usam discos para armazenamento.

Outra alternativa que não pode ser descartada é a possibilidade de gerar sua própria energia para diminuir ou até eliminar a dependência de concessionárias de energia, haja vista os resultados obtidos com o Parasol [Goiri et al. 2013], vantajoso inclusive na diminuição de emissão de carbono na atmosfera.

Caracterizar a forma como a energia está sendo consumida em um *datacenter* é importante, uma vez que todo sistema computacional consome energia para operar. A condição desejável é que o consumo seja sempre uma função proporcional à carga em processamento e claro com menor taxa de crescimento possível. *Datacenters* são dimensionados para suportar picos de demanda de processamento, mas na maior parte do tempo, trabalham com cargas bem menores. Então é natural que se desligue parcialmente recursos que não estejam sendo utilizados e que os mesmos sejam reativados assim que necessário. Partindo desse ponto de vista, desligar parte dos equipamentos em um *datacenter* é uma forma direta de se atuar no escalonamento de recursos. Trabalhos nessa linha podem ser encontrados, atuando de várias formas sobre escalonamento de recursos com sucesso. Um interessante trabalho desenvolvido em economia de energia procura tirar proveito de *datacenters* distribuídos geograficamente [Le et al. 2011]. Nele um estudo considerou que existem tarifas de energia diferenciadas ao longo de um dia, e que *datacenters* apresentam considerável consumo com refrigeração, e que as temperaturas dos ambientes geograficamente distribuídos também variam. Reunindo estes fatores, um modelo foi estabelecido para caracterizar seu comportamento. Partindo de *traces* reais, o modelo obtido nos levantamentos feitos foi simulado para várias políticas de distribuição de tarefas, inclusive com a antecipação da refrigeração requerida em função da carga, sempre fugindo do horário de pico. Consideráveis reduções de custo foram alcançadas neste desenvolvimento, demonstrando que escalonar tarefas geograficamente pode custar menos, e energia foi economizada ao se levar em conta a diminuição de demanda por refrigeração.

Outro interessante sistema, o GreenNebula [Berral et al. 2014], foi desenvolvido como uma extensão do OpenNebula para gerenciar máquinas virtuais que "seguem as energias renováveis". Tomando como recurso a migração entre *datacenters* que dispõem de energia verde, máquinas virtuais são deslocadas para onde há energia disponível, com comprovado baixo custo de migração. Para decidir a melhor alocação da máquina virtual, o sistema e suas variáveis são modelados usando programação linear.

Trabalhos desta natureza reforçam o fato de que escolher um escalonamento ade-

quado à tarefa é um bom caminho para o processamento de dados massivos visando economizar energia. As cargas deste tipo muitas vezes são cargas adiáveis [Goiri et al. 2013], ou seja podem ter seu processamento protelado (até certo ponto) para serem executadas em momentos mais adequados. Isso favorece ajustá-las ao consumo de energia renovável, ou pelo menos, usar energia de menor custo.

Caminhando mais na direção de processamento de dados massivos, no trabalho de [Leverich and Kozyrakis 2010], um *cluster* Hadoop é submetido a um experimento em que conjuntos de nodos vão sendo desabilitados para que se consiga realizar uma determinada tarefa e seu desempenho seja avaliado assim como seu consumo. Hadoop tipicamente utiliza um sistema de arquivos distribuído, o HDFS (Hadoop File System) que distribui dados através de discos em um *cluster* tirando proveito de processar dados em função de sua localidade. Oferece uma estrutura robusta ao permitir a replicação de seu conjunto de dados, visando contornar falhas comuns ao processar *big data*. HDFS garante que ao armazenar réplicas nunca duas cópias são alocados em um mesmo nó e no mínimo são mantidas duas cópias em *racks* diferentes. Com uma modificação na estrutura original do HDFS, os autores desse trabalho estabeleceram um subconjunto de cobertura que permitiu ir desabilitando nós de processamento sem comprometer a conclusão da tarefa, pela exclusão de nós que estivessem fora do conjunto de cobertura. Chama a atenção nos resultados que a medida que se suprimem nós, o tempo de computação aumenta, mas a quantidade economizada de energia é significativa, variando entre 9 e até 50 %.

Há experimentos que visam melhorar o desempenho do sistema sem necessariamente aumentar na mesma proporção o consumo de energia. No estudo feito por [Wirtz and Ge 2011] a adoção de políticas de localização de dados adequada é combinada à variação de frequência de trabalho dos processadores servidores para gastar menos energia.

Outras formas já experimentadas para economizar energia em *big data* também passam por escalonar recursos. No desenvolvimento realizado por [Mashayekhy et al. 2014], um escalonador de "consumo consciente" para tarefas MapReduce é modelado como um problema de otimização por programação inteira, em que uma função objetivo busca minimizar o consumo de energia ao alocar tarefas *Map* e tarefas *Reduce* em *slots* dentro de um *cluster*, seguindo restrições determinadas pelo tempo de resposta da tarefa assim como obedecendo a característica de não iniciar a fase de *Reduce* antes que todas as tarefas de *Map* estejam concluídas. Como solução para o problema dois algoritmos gulosos que implementam heurísticas escalonaram os recursos e conseguiram resultados de até 40 % de economia, quando comparados com o escalonamento tradicional que minimiza o *makespan*, ficando próximo de uma solução ótima.

3. Metodologia de trabalho

Para o desenvolvimento desse trabalho foi escolhido o ambiente de processamento de dados massivos Apache Spark versão 1.5.2 e o sistema de arquivos Hadoop/HDFS v2.6.0 sobre Yarn. Com a finalidade de realizar as medições, foi adquirido, instalado e configurado um PDU (*Power Distribution Unit*) Raritan PX-2 da série 5000, com exatidão de 1% (norma ISO/IEC 62053-21), que é uma unidade de distribuição de energia capaz de monitorar variados parâmetros elétricos. Em nosso trabalho medimos a potência ativa por tomada, que corresponde ao consumo de cada servidor.

A orquestração dos servidores virtualizados foi feita pelo OpenStack v2.3.1 com sua estrutura montada em uma máquina servidora core-i7 2600 3,4GHz com 16 GB de memória RAM. Para os nós de computação foram usadas 6 máquinas servidoras Intel Xeon E5-2620v4 2,1GHz com 32 GB de RAM, duas interfaces de rede Gigabit Ethernet e disco sata de 2 TB sobre os quais foi possível estabelecer variadas configurações de HDFS. Todos os dados produzidos de consumo de energia, uso de CPU, disco, ocupação de memória, carga da máquina e tráfego de rede, além dos *logs* do processamento do Spark e da utilização do HDFS foram monitorados e registrados em um outro *cluster* de monitoração *Seshat*, que contou com mais três servidores também core-i7 2600 3,4GHz com 16 GB de memória RAM. O objetivo é que a partir da coleta sistemática de dados de experimentos, fosse possível avaliar o comportamento de diferentes políticas para o escalonamento das tarefas de processamento de dados massivos, principalmente visando o consumo de energia elétrica mais eficiente.

Como carga foram processados 8 GB de dados contendo um conjunto de *posts* do Twitter, contendo *tweets* coletados usando a API da plataforma, em busca de padrões frequentes via algoritmos Twidd e Eclat em implementação paralela que foi distribuído em servidores virtualizados sobre KVM. A aplicação Twidd é uma implementação do algoritmo FPGrowth sobre a abstração de RDD's (Resilient Distributed Datasets) e resolve o problema de mineração de padrões frequentes, no qual dada uma série de transações, cada uma composta por um conjunto de itens, e um limiar, ele encontra todos os subconjuntos de itens que ocorrem no mínimo a quantidade de vezes especificada por esse limiar. A motivação para o uso desta aplicação no trabalho reside no grande custo computacional inerente ao processo de geração dos subconjuntos de itens para cada transação, o que consequentemente requer uso intenso do processador e se reflete em um maior uso de energia [Dias et al. 2016]. O Eclat é também um algoritmo para mineração de padrões frequentes, mas que trabalha sobre uma representação vertical da base de dados. Ao invés de um conjunto de itens para cada transação, temos conjuntos de identificadores de transações para cada item. Eclat é iterativo pela sua característica hierárquica durante a mineração e irregular pela combinatória envolvida na distribuição dos *itemsets* frequentes [Dias et al. 2016]. Dessa forma, os procedimentos realizados a cada iteração são os mesmos, mas os dados de entrada/saída variam e consequentemente seus padrões de comunicação também são variáveis, tornando interessante seu estudo sobre o ponto de vista de consumo energético. Para ambos os algoritmos e em função da base de dados adotada, foram definidas 128 partições, com suporte mínimo de 0,02.

3.1. Arquitetura de monitoração

Para monitorar os experimentos foi construída uma estrutura de coleta de métricas e *logs*. Como o foco desse trabalho é voltado a medir o consumo de energia sua representação esquemática pode ser vista na Figura 1, em que se destaca o PDU. A partir de suas tomadas, o PDU distribui e mede individualmente a energia elétrica que é entregue para as máquinas físicas do *cluster* que estabelece um ambiente de nuvem de processamento. Este PDU é capaz de medir tensão, corrente, fator de potência, individualmente para cada tomada, mas a grandeza de maior relevância para esse estudo é a potência ativa consumida pelos servidores e ativos de rede, uma vez que representa a energia que efetivamente realiza trabalho.

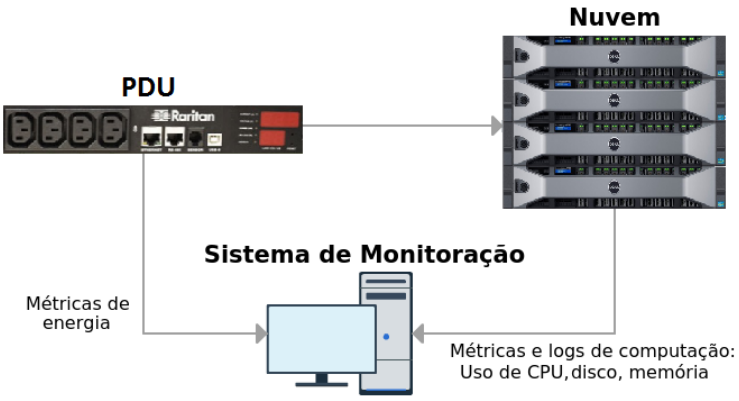

Figura 1. Esquema físico de coleta.

Nessa nuvem as aplicações que são executadas fornecem ao sistema de monitoração diversas métricas como tempo de execução, uso de CPU, utilização de memória, utilização de discos, tráfego de rede, entre outras. O registro de *logs* é capaz de coletar e registrar mensagens de execução de tarefas Spark, assim como do Yarn e do HDFS.

Para a coleta das informações providas tanto pelos servidores físicos e virtualizados assim como pelo PDU, um *cluster* constituído de três máquinas, armazenam e processam para tornar adequadas à visualização as métricas e os *logs*, gerados na nuvem. A arquitetura adotada para monitorar escolhida foi a *Seshat* que integra vários *softwares open-source*. Para coletar métricas extraídas dos servidores o *framework* Sensu foi instalado, seguindo a arquitetura e combinado ao *daemon* Collectd que é mais adequado para recuperar medições feitas pelo PDU que se comunica via SNMP. Para o armazenamento de métricas a ferramenta Influxdb foi escolhida por ser um banco de dados otimizado para o tratamento de séries temporais. Compõe ainda a arquitetura uma ferramenta que permite a visualização dos dados, o Grafana. Com ela é possível visualizar métricas de aplicações que estejam em execução na nuvem e que são registradas em um banco de dados. Esta ferramenta permite a criação de *dashboards* de maneira bastante flexível e conta com elevado poder de configuração de formatos de exibição. No caso deste sistema o Grafana é usado para plotar o comportamento das variáveis medidas no PDU e extraídas simultaneamente do Sensu, enquanto as aplicações Spark estão em execução na nuvem.

Ainda nessa arquitetura há agentes instalados nos servidores que são capazes de monitorar *logs* de execução, que ficam armazenados no Elastic Search que é um banco de dados mais adequado para *logs* e que trabalha associado ao Kibana, uma poderosa ferramenta de visualização e tratamento de *logs*.

Ao executar aplicações, é possível também rastrear variados detalhes da execução da tarefa, tais como tempo gasto em cada etapa, resultados intermediários de processamento, tudo por meio do *History Server* disponibilizado pelo próprio Spark. Desta forma cada vez que uma aplicação é disparada no *cluster* é possível a partir de seu *timestamp* buscar no banco de dados a energia que foi consumida durante sua execução. Uma vez que o PDU mede a potência ativa, e via monitoração a armazena no Influxdb, podemos determinar a quantidade de energia consumida para cada execução de uma determinada

aplicação.

3.2. Abordagem do problema

Ao executar aplicações no ambiente Spark, um considerável conjunto de parâmetros pode ser variado. Como é uma aplicação distribuída, Spark nos permite definir parâmetros de configuração para o mestre (também chamado de *driver*) assim como para seus escravos (normalmente conhecidos como executores), além de ser possível escolher a quantidade de partições que serão feitas sobre o conjunto de dados. Em nossos experimentos, um primeiro conjunto de testes nos permitiu definir a configuração de valores de *setup* para adequar o ambiente Spark ao processar dados massivos em duas tarefas de mineração de padrões, Twidd e Eclat quando executado sobre máquinas virtuais (VMs). Feito isso, nossos testes avaliaram sistematicamente o impacto de variar a disponibilidade de núcleos (*cores*) de processamento para diferentes quantidades de executores em duas políticas de ocupação física da nuvem. Os passos para isso:

- Foi estruturado um ambiente de testes Spark contando com o Yarn como escalonador de recursos para executar Twidd e Eclat sobre uma base de dados de 8 GB distribuídas em um sistema de arquivos HDFS nos nós de computação, sempre com fator de replicação de dados 3 formando a nuvem para experimentação. O suporte mínimo dos algoritmos foi o mesmo e definido em 0,02 para garantir um efeito de carga significativa sobre a nuvem.
- Foram testadas duas formas de ocupar progressivamente os nós de computação. O HDFS foi montado múltiplas vezes, acrescentando VM's na correspondência que agregasse 16 *cores* para cada novo experimento. Cada experimento citado executou cada uma das aplicações com replicação 10, o que é considerado um valor adequado a experimentos com pouca variação.
- Três configurações de VM's diferentes foram escolhidas para ocupar progressivamente o *cluster*, com objetivo de avaliar o impacto da granularidade da virtualização utilizada, tanto para a efetiva conclusão das tarefas quanto para seu consumo. Na primeira configuração de experimentos, foram disparadas V.M.'s de 4 *cores*. Na segunda, foram usadas VM's de 8 *cores* e na terceira e última VM's de 16 *cores* ocupavam completamente cada um dos servidores físicos.
- Duas políticas de ocupação dos recursos, tendo como foco o uso de *cores* de processamento disponíveis, foram avaliadas. A primeira usou uma estratégia gulosa, onde cada servidor de computação foi ocupado por um HDFS montado e os experimentos executados. A cada passo guloso seguinte outro servidor físico foi incluído e novamente os experimentos foram executados. Desse modo, sempre foram acrescidos 16 *cores* físicos a cada passo. Na segunda estratégia, de forma similar a um escalonamento Round-Robin, os acréscimos de *cores* para montagem e testes de experimentos foram feitos exatamente de forma ortogonal à estratégia anterior, já que acréscimos também de 16 *cores* físicos eram feitos só que sempre ocupando servidores físicos da forma mais distribuída.

  Os resultados encontrados estão apresentados na próxima seção.

4. Testes e discussão de resultados

Em nosso trabalho, a avaliação de dois parâmetros foram foco: **consumo de energia** (medido em watthora) e **desempenho** ao executar as aplicações na forma de tempo para

concluir a tarefa (medido em segundos). Reforçamos que o consumo de energia possui uma relação atrelada ao tempo de execução da tarefa. Uma expectativa natural é que tarefas que sejam concluídas mais rapidamente apresentem tendência de consumir menos, uma vez que o consumo da estrutura sem carga é bem significativo. Seguindo a metodologia que foi apresentada na seção anterior obtivemos resultados que estão apresentados nos gráficos que compõem as Figuras 2 e 3.

Os gráficos mostram a execução dos dois algoritmos, Twidd e Eclat, avaliados em tempo de conclusão da tarefa em função da disponibilidade de *cores* e consumo ao executar a tarefa em função da mesma disponibilidade de *cores*. Como a quantidade de *cores* foi avaliada em saltos de 16 *cores* ocupando fisicamente os seis servidores de forma gulosa ou em Round Robin, temos em cada gráfico 5 séries que são executadas em 6 configurações diferentes, definindo os pontos apresentados. São 5 séries porque na terceira configuração de V.M.'s, com 16 *cores*, a ocupação da nuvem na política de Round Robin acaba se tornando a mesma da forma gulosa. Também com V.M.'s de 16 *cores* não permite que exista o primeiro ponto, uma vez que o *driver* do Spark necessitará de ao menos um executor, o que acaba ocupando pelo menos 32 *cores*. As linhas presentes em cada série do gráfico tem o objetivo apenas de facilitar a interpretação do mesmo. Todos os valores apresentados nos experimentos contam com uma confiança de 95% sobre as médias. Os valores absolutos das confianças encontrados nunca foram maiores que 4,7% de suas respectivas médias, sendo tipicamente bem inferiores. Para evitar excesso de informações nos gráficos, as confianças foram suprimidas evitando poluição visual.

De início, nos chama atenção que uma tarefa executada na mesma base de dados pode apresentar resultados de desempenho e consumo muito distintos ainda que utilizando da mesma estrutura física. Em valores, podemos afirmar a partir dos gráficos que executar Twidd distribuído pode demorar entre 600 segundos (16 *cores* em V.M.'s de 4 *cores* distribuídas de forma gulosa) e 110 segundos (96 *cores*, nuvem totalmente ocupada), conforme a quantidade de *cores* disponível. Para executar o Eclat distribuído, os tempos estão entre 1200 segundos (16 *cores* em V.M.'s de 4 *cores* distribuídas de forma gulosa) para o pior resultado e 123 segundos (48 *cores* em V.M.'s de 8 *cores* cada distribuídas de forma gulosa) no melhor caso, que são diferenças significativas, mas esperadas uma vez que se disponibilizou recursos de forma crescente para cada ensaio.

Por outro lado, no consumo de energia para executar as mesmas tarefas, podemos ter variações que vão de 14 wh (48 *cores* em V.M.'s de 8 *cores* distribuídas de forma gulosa) a 36 wh (32 *cores* em V.M.'s de 4 *cores* distribuídas de forma gulosa) para o Twidd. No Eclat as variações ficam entre 16 wh (16 *cores* em V.M.'s de 8 *cores* distribuídas em Round Robin) e 49 wh (48 *cores* em V.M.'s de 8 *cores* distribuídas de forma gulosa). Isso nos permite afirmar que não é uma tarefa trivial a decisão sobre qual forma de escalonar recursos em um ambiente virtualizado para processar dados massivos que apresente bom compromisso entre o consumo de energia e tempo de execução.

A observação dos gráficos mostra que o aumento de *cores* em muitos casos não traz ganhos significativos, então seria interessante achar soluções que encontrassem as configurações que representem maior economia em tempo ou consumo sem exigir muitos recursos, para não elevar muito o custo. De qualquer forma há relações entre o tempo e consumo que precisam ser avaliados posteriormente.

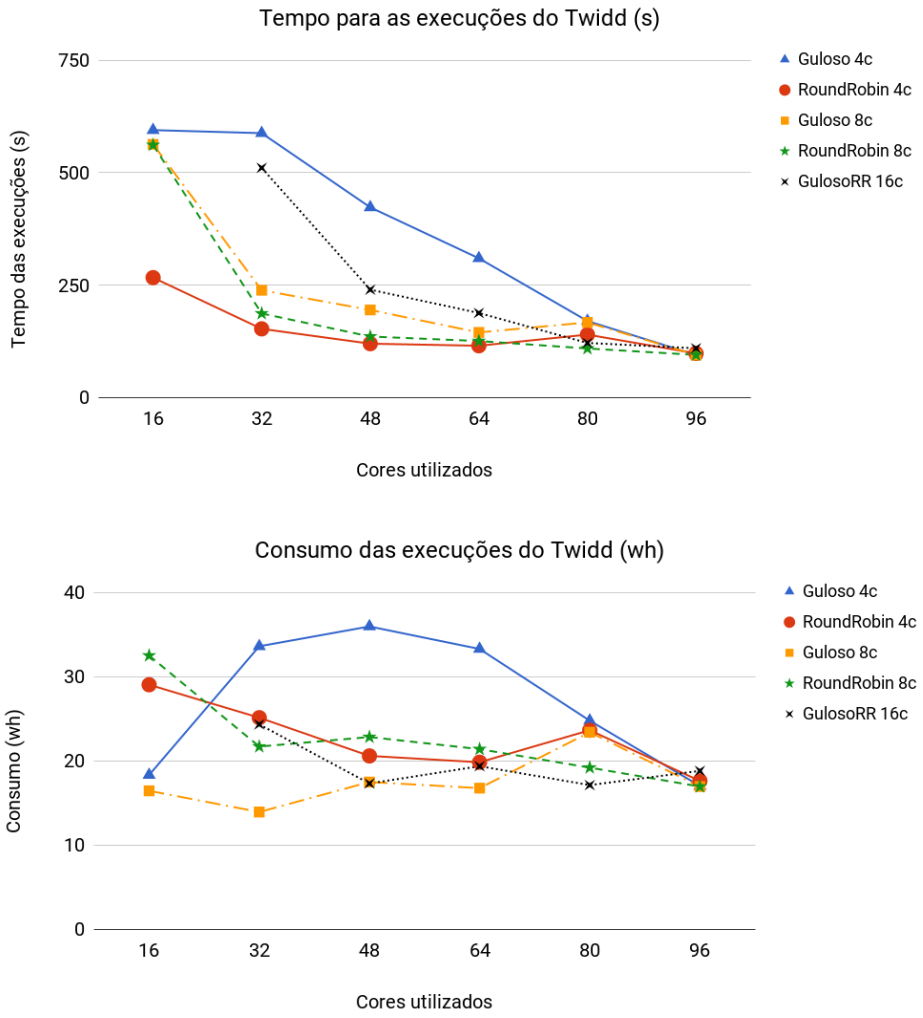

Figura 2. Gráficos do Twidd em função do tempo de execução e em função do consumo para variadas disponibilidades de *cores* em duas políticas de escalonamento: Gulosa e Round Robin

Do ponto de vista do desempenho, à medida que se oferece mais *cores* para a conclusão da tarefa, era esperado que o tempo de execução diminuísse. Nas curvas de tempo de execução para o Twidd e para o Eclat, na maioria das vezes o tempo diminui a medida que mais *cores* são oferecidos para a tarefa. Porém a forma como esse tempo diminui apresenta pouca inclinação e por vezes o tempo de execução volta a aumentar conforme pode ser observado para o Twidd entre os pontos de 64 e 80 *cores* para para V.M.'s com 8 *cores* na ocupação gulosa e também para V.M.'s de 4 *cores* em Round Robin. Para o Eclat o desempenho piora quando se oferece mais recurso apenas para V.M.'s de 4 *cores* em Round Robin entre os pontos de 64 para 80 *cores*. Essas alterações acabam por apresentar um reflexo também no consumo de energia para a tarefa, se observarmos a correspondência com os mesmos pontos nos gráficos de consumo.

Nuvens tipicamente seguem a política gulosa de ocupação das V.M.'s. Nos chama a atenção o fato de que V.M.'s de 4 *cores* que ocupam a nuvem na política Round Robin apresentam desempenho significativamente melhor, em ambos os algoritmos. Para o

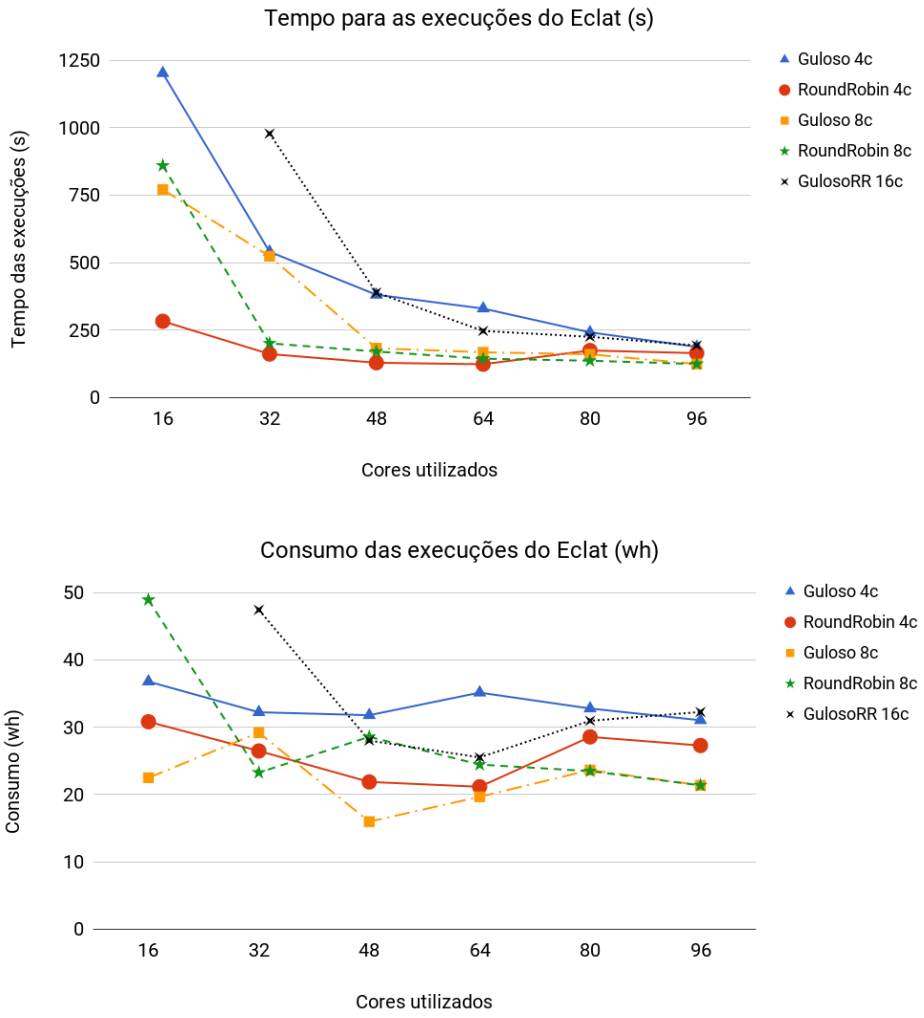

Figura 3. Gráficos do Eclat em função do tempo de execução e em função do consumo para variadas disponibilidades de *cores* em duas políticas de escalonamento: Gulosa e Round Robin

Twidd, o impacto disso é significativo mas para o Eclat que é um algoritmo mais simples, praticamente leva o consumo para valores bem inferiores, só não consumindo menos que alguns dos pontos em que a nuvem foi ocupada de forma gulosa por V.M.'s de 8 *cores*. Se considerarmos apenas V.M.'s de 4 *cores* para o Twidd, a diferença de consumo é inegavelmente favorável à ocupação em Round Robin. O uso de Round Robin é normalmente melhor em termos de tempo, pois espalha as máquinas distribuindo mais a carga.

Entretanto, é interessante notar também que a ocupação da nuvem por V.M.'s de 8 *cores* na estratégia gulosa alcançou os melhores resultados em consumo de energia, em praticamente todas as situações. Mesmo nas situações onde não foi a melhor opção, ainda esteve próxima dos melhores resultados.

Uma avaliação do início de todas as curvas também nos permite perceber que trabalhar com poucos recursos (*cores*), faz com que tenhamos um desempenho ruim seja para tempo de execução ou o consumo de energia. Configurações com poucos *cores* e

consequentemente menos memória, geram os piores tempos, pois é mais difícil conter o problema todo em memória. Isso é ainda pior para V.M.'s maiores porque perde-se mais *cores* na máquina *master* e porque um nó que pare por conta de *garbage collection* tem mais impacto, pois irá parar mais *cores* de uma vez. Justamente esse fato pode ser uma explicação para o fato de a política de ocupação com V.M.'s em Round Robin ser tão melhor: há menos contenção de recursos e quando algum nó para no *garbage collector*, o volume de recursos congelado é menor.

5. Conclusão e Trabalhos Futuros

Visando avaliar o consumo de energia ao processar dados massivos em ambientes virtualizados, estabelecemos um ambiente de nuvem amplamente monitorado para avaliarmos o impacto do escalonamento de recursos em função do tamanho, quantidade e políticas de ocupação física da estrutura. Nele foram realizados testes com dois algoritmos paralelos de execução distribuída que nos permitiram perceber que tanto o desempenho mas principalmente o consumo não possuem um comportamento simples para que se defina a melhor maneira de distribuir tarefas Spark gastando menos energia. Os experimentos realizados evidenciaram que há muito que ser estudado uma vez que o mesmo problema executado em configurações diferentes de recursos pode apresentar diferenças no consumo de energia que sejam mais que o dobro entre elas. Também foi possível perceber que a forma típica de ocupar um *datacenter* que é Gulosa, justamente para permitir o desligamento de servidores para economizar energia em cargas menores, nem sempre representa a melhor realidade, quando se trata de processar dados massivos em ambientes virtualizados.

Como trabalhos futuros, os testes realizados nos sugerem experimentar novos algoritmos de processamento de dados massivos com características que se diferenciem dos aqui utilizados, para que seja possível avaliar um comportamento mais completo das cargas típicas para esses ambientes. Outros virtualizadores além do KVM devem ser considerados para testes, para verificar se a validade do que encontramos aqui tem a mesma correspondência.

Dentro do objetivo de estudar o ambiente, também pode-se considerar variar a forma como se aproveita da localidade de dados via novas maneiras de estabelecer o HDFS, ou mesmo o uso de ambientes de armazenamento diferentes para verificar se é possível economizar energia. Também acreditamos ser interessante avaliar se é possível tirar maior proveito dos nós de computação pelo uso do Driver do Spark fora do HDFS, uma vez que pode desocupar mais uma V.M. para dedicá-la ao processamento da tarefa efetivamente, o que consequentemente deve ter impacto no consumo de energia.

Agradecimentos

Este trabalho foi parcialmente financiado por Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-14), H2020-EUBR-2015 EUBra-BIGSEA e H2020-EUBR-2017 Atmosphere.

Referências

Abts, D., Marty, M. R., Wells, P. M., Klausler, P., and Liu, H. (2010). Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347.

Andersen, D. G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., and Vasudevan, V. (2009). Fawn: A fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14. ACM.

Berral, J. L., Goiri, Í., Nguyen, T. D., Gavalda, R., Torres, J., and Bianchini, R. (2014). Building green cloud services at low cost. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 449–460. IEEE.

Conceição, V. S., Volpini, N. D. O., and Guedes, D. (2018). *Seshat*: uma arquitetura de monitoração escalável para ambientes em nuvem. In *Anais do XVII Workshop em Desempenho de Sistemas Computacionais e de Comunicaç ao, Natal-RN. Sociedade Brasileira de Computaçao (SBC)*.

Dias, V., Meira, W., and Guedes, D. (2016). Dynamic reconfiguration of data parallel programs. In *2016 28th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 190–197.

Dupont, C., Giuliani, G., Hermenier, F., Schulze, T., and Somov, A. (2012). An energy aware framework for virtual machine placement in cloud federated data centres. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–10. IEEE.

Ferro, M., Yokoyama, A., Klôh, V., Silva, G., Gandra, R., Bragança, R., Bulcao, A., Schulze, B., and SA-PETROBRAS, P. B. (2017). Analysis of gpu power consumption using internal sensors. In *Anais do XVI Workshop em Desempenho de Sistemas Computacionais e de Comunicaç ao, Sao Paulo-SP. Sociedade Brasileira de Computaçao (SBC)*.

Goiri, I. n., Katsak, W., Le, K., Nguyen, T. D., and Bianchini, R. (2013). Parasol and greenswitch: Managing datacenters powered by renewable energy. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, pages 51–64, New York, NY, USA. ACM.

Gu, X., Hou, R., Zhang, K., Zhang, L., and Wang, W. (2011). Application-driven energy-efficient architecture explorations for big data. In *Proceedings of the 1st Workshop on Architectures and Systems for Big Data*, pages 34–40. ACM.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., and Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115.

Kansal, A., Zhao, F., Liu, J., Kothari, N., and Bhattacharya, A. A. (2010). Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 39–50. ACM.

Kgil, T., D'Souza, S., Saidi, A., Binkert, N., Dreslinski, R., Mudge, T., Reinhardt, S., and Flautner, K. (2006). Picoserver: using 3d stacking technology to enable a compact energy efficient chip multiprocessor. *ACM SIGARCH Computer Architecture News*, 34(5):117–128.

Kim, K. H., Beloglazov, A., and Buyya, R. (2009). Power-aware provisioning of cloud resources for real-time services. In *Proceedings of the 7th International Workshop on*

Middleware for Grids, Clouds and e-Science, MGC '09, pages 1:1–1:6, New York, NY, USA. ACM.

Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, page 9. Disponível em http://www.analyticspress.com/datacenters.html.

Le, K., Bianchini, R., Zhang, J., Jaluria, Y., Meng, J., and Nguyen, T. D. (2011). Reducing electricity cost through virtual machine placement in high performance computing clouds. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 22. ACM.

Leverich, J. and Kozyrakis, C. (2010). On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Operating Systems Review*, 44(1):61–65.

Mashayekhy, L., Nejad, M. M., Grosu, D., Lu, D., and Shi, W. (2014). Energy-aware scheduling of mapreduce jobs. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pages 32–39. IEEE.

Ramanathan, R. M. (2006). Intel® multi-core processors. http://www.intel.com/technology/archteture/downloads/quad-core-06.pdf.

Sahoo, J., Mohapatra, S., and Lath, R. (2010). Virtualization: A survey on concepts, taxonomy and associated security issues. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 222–226. IEEE.

Shehabi, A., Smith, S. J., Sartor, D. A., Brown, R. E., Herrlin, M., Koomey, J. G., Masanet, E. R., Horner, N., Azevedo, I. L., and Lintner, W. (2016). United states data center energy usage report.

Whitney, J. and Delforge, P. (2014). Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers. https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf. Acessado em maio de 2015.

Wirtz, T. and Ge, R. (2011). Improving mapreduce energy efficiency for computation intensive workloads. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8. IEEE.

Ye, K., Huang, D., Jiang, X., Chen, H., and Wu, S. (2010). Virtual machine based energy-efficient data center architecture for cloud computing: a performance perspective. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 171–178. IEEE Computer Society.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.