Uma Análise Multicamadas do Consumo de Energia em Cargas *Big Data*

Nestor D. O. Volpini[1,3], Vinícius Dias[2], Dorgival Guedes[3]

[1]Departamento de Eletroeletrônica e Computação (CEFET-MG) – Contagem, MG

nestor@cefetmg.br

[2]Departamento de Ciência da Computação (UFLA) – Lavras, MG

viniciusdias@ufla.br

[3]Departamento de Ciência da Computação (UFMG) – Belo Horizonte, MG

{nestor,dorgival}@dcc.ufmg.br

Resumo. *Datacenters consomem 1% de toda a energia mundial. Este trabalho estudou os fatores que compõem o consumo de energia ao processar big data sobre Spark. Caracterizou os fatores mais significativos a partir de um conjunto de testes em cargas big data, que foram processadas com a energia medida fim a fim agregada a métricas da infraestrutura. Um estudo multifatorial sobre os resultados, demonstrou que: (i) nem sempre o acréscimo de recursos acelera o processamento a ponto se economizar energia; (ii) a forma como o recurso é ofertado (tamanho de VM) também altera o desempenho e o consumo; (iii) uma monitoração integrada a uma abordagem multicamada é fundamental para elencar fatores que podem ser a chave na economia de energia.*

1. Introdução

Considerando os recursos tecnológicos presentes na vida moderna, é possível observar um crescimento significativo na geração de dados que demandam processamento para torná-los relevantes. Redes de sensores, dispositivos vestíveis pessoais, hardwares conectados por Internet das Coisas são alguns exemplos do elevado volume de dados que tem sido gerado. Por outro lado, sistemas de recomendação, algoritmos de aprendizado de máquina, mineração de dados e outras recentes evoluções em inteligência artificial se beneficiam desses dados tornando-os relevantes para algum contexto. Para processar esse conjunto de dados massivos é necessário contar com uma infraestrutura não trivial. Nuvens computacionais tornam esse tipo de trabalho exequível e acessível, principalmente por usar os recursos com um custo proporcional à sua utilização, eliminando grandes investimentos iniciais de seus usuários [Armbrust et al. 2010]. A computação em nuvem, ao ofertar recursos de forma virtualizada reduz custos para o provedor e para o usuário, flexibilizando seu uso de maneira escalável, contando com elevada disponibilidade de serviços. O uso de virtualização para processar grandes volumes de dados se consolida pelo uso de Máquinas Virtuais (VMs do inglês *Virtual Machines*) desacoplando o processamento da parte física permitindo ao usuário alocar um determinado volume de recursos (núcleos, memória, discos etc.) da forma que lhe convier [Assunção et al. 2015].

Por outro lado, o consumo de energia demandado por esse tipo de infraestrutura é significativo ao operar grandes volumes de dados. Seu custo é relevante tanto para

os provedores de nuvens como para o meio ambiente, já que muitas vezes advém de uma matriz de produção energética que se sustenta emitindo preocupantes quantidades de carbono na atmosfera [Whitney and Delforge 2014]. De acordo com o IEA (International Energy Agency) [IEA 2022], no ano de 2022, *datacenters* consumiram 1% de toda a energia produzida no mundo, o que é uma fatia considerável com previsão de chegar a 8% do total até o final da próxima década [Pesce 2021]. Medir adequadamente o consumo de energia requer uma visão global dos processos de computação, o que é complexo [Anand et al. 2022]. A título de exemplo, pode-se comprimir dados para diminuir o volume do tráfego em uma rede, levando a um menor consumo em seus ativos. Porém, há o incremento na energia consumida pelos processadores ao comprimir e descomprimir os dados que precisa entrar no balanço. Para aferir uma redução de consumo realizada por uma determinada intervenção, é necessário medir a energia consumida de ponta a ponta.

Desta forma, visando apurar e reduzir o consumo energia, monitorando processos de uma ponta a outra, este trabalho avaliou a execução de cargas de grandes volumes de dados em um ambiente de porte reduzido, similar a uma nuvem. Duas medidas essenciais fundamentaram os resultados encontrados: desempenho traduzido em tempo de conclusão das tarefas, e consumo total de energia para cada tarefa. O estudo buscou seus resultados obtidos à partir de experimentos, em que foram estabelecidos os seguintes objetivos: **[OBJ-I]** definir um melhor conjunto de recursos para cada carga, em função das medições; **[OBJ-II]** verificar se o tamanho das VMs pelas quais são ofertados os recursos impactam nos resultados; e **[OBJ-III]** estabelecer modelos que expliquem as medidas de tempo e o consumo de energia. Como cargas, foram utilizadas nos experimentos cinco implementações de algoritmos de *big data*.

2. Trabalhos relacionados

O trabalho de [Leverich and Kozyrakis 2010] inspirou diversos estudos sobre o consumo de energia em *datacenters* sobre cargas *big data*. Há oportunidade para se economizar energia pela escolha de uma arquitetura [Gu et al. 2011] ou movimentando VMs para localidades onde há energia renovável como em GreenNebula [Berral et al. 2014]. É possível atuar na topologia do *datacenter* e adequá-la ao consumo [Abts et al. 2010], e ainda encontrar a melhor rota de comunicação [Baker et al. 2015]. Há também estudos que atuam na tensão e frequência de processadores (DVFS) [Kim et al. 2009] e há trabalhos que economizam energia alocando tarefas a recursos [Mashayekhy et al. 2015], até mesmo no nível de "federação de *grids* computacionais" [Forte et al. 2018]. Há casos em que se busca economia ao processar algoritmos em processadores ARM [Bernardo et al. 2021], mas isso requer investimentos significativos, enquanto nosso estudo busca aproveitar a infraestrutura que tipicamente já se tem disponível. Já há inclusive, atenção dirigida ao consumo em processar IA (Inteligência Artificial) [Bernardo et al. 2020].

Neste trabalho, estudou-se o desempenho e o consumo de aplicações em *big data* sobre Spark, dada sua ampla utilização [Saeed et al. 2020]. Trabalhos como o de [Li et al. 2020] e de [Gonçalves et al. 2023] atuaram em escalonamento, o primeiro distribuindo tarefas em nuvens e o segundo definindo para qual núcleo de processamento irá uma tarefa. Como contribuição, foi feita a opção por medir o consumo de energia de ponta a ponta para evitar vieses que possam surgir de resultados que não considerem o sistema como um todo [Anand et al. 2022], o que comprovou a importância das

recomendações feitas por [Ousterhout 2018]. Além disso, houve uma contribuição com uma análise multicamada dos resultados que, até onde se sabe, é uma abordagem diferenciada para esse tipo de estudo.

3. Metodologia

Inspirado pelo estudo feito em [Volpini et al. 2018], este trabalho avaliou os fatores determinantes para compor o tempo de execução e o consumo de energia de cinco algoritmos de processamento massivo de dados implementados sobre o *framework* Spark (seção 3.1). Aqui a avaliação foi realizada sobre uma estrutura de monitoração projetada especificamente para coletar e integrar métricas em nível de *hardware*, sistema operacional e aplicação (seção 3.2). A partir dos dados coletados nas execuções, foi elaborada uma análise muito mais aprofundada e detalhada do comportamento das aplicações frente aos fatores relevantes ao consumo de energia (seções 4, 5 e 6).

3.1. Cargas avaliadas

Todas as cargas utilizadas nos testes tiveram suas configurações de execução ajustadas de forma iterativa antes da realização dos testes. Durante esse processo de calibração, foram seguidas as boas práticas recomendadas pela documentação do Spark [Spark 2015]. Para as cargas em que o particionamento não foi configurado automaticamente, o número de partições foi determinado conforme as diretrizes estabelecidas nas boas práticas.

1. TeraSort [Higgs 2018]: implementa um algoritmo de ordenação de dados sobre a abstração de RDD's (*Resilient Distributed Datasets*). O interesse por essa aplicação em nossos testes foi por uso intenso de embaralhamento (*shuffle*), o que notadamente é um dos gargalos em desempenho para esse tipo de processamento. A complexidade do Terasort é dominada pela fase de ordenação, sendo da ordem de $O(n \log(n))$.

2. K-means [Li et al. 2017]: é um algoritmo de agrupamento de ampla utilização em mineração de dados. A base de dados sobre a qual atuou contou com 100 milhões de linhas em 24 colunas. São limitadores de seu desempenho a quantidade de memória principal e a disponibilidade de CPU. A complexidade do K-Means é dependente do número de pontos dados (N), da quantidade de *clusters* desesejada (K), a dimensão (D) dos dados e da quantidade de iterações (T) necessárias para convergir. Portanto, a complexidade total do K-Means é dada por $O(NKDT)$.

3. PageRank [Li et al. 2017]: é um algoritmo que estima a importância de nós em uma rede (por exemplo, de páginas da Web), caracterizado por sua implementação distribuída e iterativa. Sua complexidade depende de calcular a importância de cada nó de um grafo com base na importância de outros nós que apontam para ele. A complexidade de tempo para cada iteração é $O(E)$, sendo E é o número de arestas na rede/grafo.

4. Support Vector Machine (SVM) [Li et al. 2015]: é um algoritmo de aprendizado de máquina supervisionado, usado principalmente para tarefas de classificação e regressão. O objetivo do SVM é encontrar o hiperplano que melhor separa os dados, maximizando a distância entre os pontos mais próximos de cada classe. A complexidade inerente desse algoritmo advém da inversão de matrizes, chegando a alcançar $O(n^3)$.

5. Matrix Factorization (MatFact) [Li et al. 2017]: é um algoritmo amplamente utilizado em sistemas de recomendação e análise de dados, decompondo matrizes em fatores menores que capturam a estrutura subjacente dos dados. A fatoração é implementada pelo

algoritmo *Alternating Least Squares* (ALS). ALS é um algoritmo que a cada iteração resolve um problema de mínimos quadrados para atualizar as matrizes fatoriais. A complexidade por iteração é geralmente $O(MNK)$, onde M e N são as dimensões das matrizes originais e K é o número de fatores, tipicamente as dimensões da decomposição.

A tabela 1 apresenta as características das cargas avaliadas, demonstrando a quantidade de estágios em cada uma e o volume de dados efetivamente processados, em tarefas críticas. As características destacadas contribuíram para gerar uma diversidade de comportamentos capaz de estressar o ambiente de processamento. Esses valores foram extraídos dos próprios *logs* do Spark.

Tabela 1. Resumo das cargas utilizadas no estudo experimental.

Carga	Qtde. de estágios	Entrada	Saída	Leitura *Shuffle*	Escrita *Shuffle*
Terasort	2	≈29 GB	≈29 GB	≈15 GB	≈16,5 GB
K-means	15	≈16 GB	≈19 GB	≈4 MB	≈4,3 MB
PageRank	98	≈2,6 GB	≈110 MB	≈7,8 GB	≈9,4 GB
SVM	28	≈16,7 GB	≈30,8 MB	≈13,2 GB	≈14,4 GB
MatFact	66	≈2,6 GB	≈2,8 GB	≈6,7 GB	≈7 GB

3.2. Ambiente de execução e monitoração

A figura 1 ilustra o ambiente de execução e monitoração utilizado. Em linhas gerais, parte do recurso físico foi separado para monitorar métricas de desempenho e consumo, e outra parte designado para a execução das aplicações. Essa separação foi feita de modo a minimizar o impacto da monitoração nas execuções monitoradas.

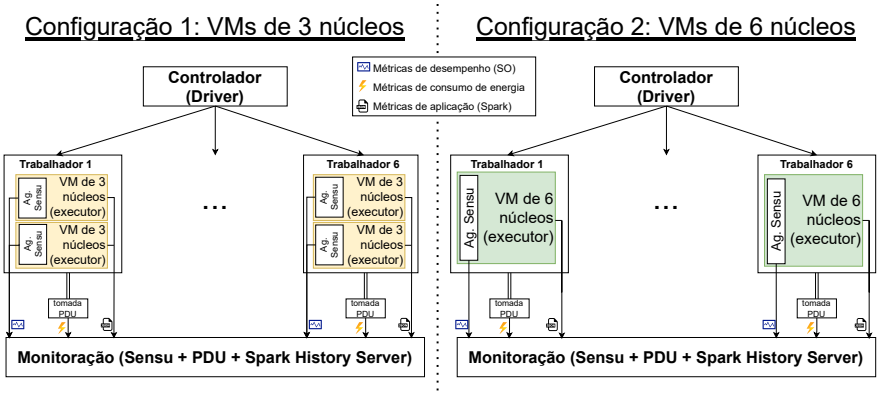


Figura 1. Esquema do ambiente monitorado de execução das cargas.

Ambiente de execução. As aplicações foram executadas na plataforma Apache Spark v2.2.0, utilizando o sistema de arquivos Hadoop/HDFS v2.9.1 com replicação dois, para oferecer replicação mínima e conferir agilidade em escrita e rede. O tamanho de unidade de alocação utilizada foi de 128 MB. O gerenciador de recursos utilizado foi o Yarn, já disponibilizado junto com o Spark. Para provisionamento e orquestração do ambiente de máquinas virtuais utilizamos OpenStack na versão Pike em um servidor físico Intel Core-i7 2600 3,4 GHz com 16 GB de RAM. Para as máquinas virtuais (VMs) foram utilizados 6 servidores Intel Xeon E5-2620v4 2,1 GHz com 8 núcleos (16 *threads*) e 32 GB de RAM, duas interfaces de rede Gigabit Ethernet e um disco SATA de 2 TB, contando com o *hypervisor* QEMU-KVM 2.10.1 sobre GNU/Linux Ubuntu 16.04, Kernel 4.4.0.

Configuração dos testes. Para os testes não foram disponibilizados todos os núcleos de processamento, nem toda a memória RAM disponível em cada servidor, para estabelecer uma reserva para o virtualizador [Asyabi et al. 2018]. Visando o **[OBJ-I]**, foram ofertados de 3 até 6 servidores físicos, acrescentando um por vez, o que correspondeu a oferecer de 18 a 36 núcleos em incrementos de 6 núcleos por vez. Do ponto de vista de consumo, é importante destacar que toda vez que uma máquina física é incluída no processamento, há um significativo acréscimo de consumo de energia P_0. Uma vez que o consumo é dado pela integração da potência ao longo do tempo, a questão a ser respondida foi: *a aceleração obtida para conclusão da tarefa justificou o acréscimo no consumo total?* Visando **[OBJ-II]**, os testes realizados contaram ou com uma VM para cada servidor físico com 6 núcleos e 30 GB de memória (1x6), ou com duas VMs de 3 núcleos em cada servidor físico, cada uma contando com 15 GB de memória RAM (2x3). Com isso, mantendo a mesma quantidade de recurso em quantidade de servidores avaliamos o impacto de se usar mais (2x3) ou menos (1x6) VMs. Cada carga (seção 1) foi executada 20 vezes para cada configuração de quantidade de servidores e tamanho de VM. Os resultados apresentados a seguir são valores médios, com seus respectivos desvios estatísticos. A cada execução as *caches* foram devidamente esvaziadas.

Ambiente de monitoração. Para monitorar potência ativa, tensão, corrente e o consumo de cada servidor utilizamos a unidade de distribuição de energia PDU (*Power Distribution Unit*) Raritan PX-2 da série 5000, com exatidão de 1% (norma ISO/IEC 62053-21). Com a finalidade de registrar os dados de utilização dos processadores, uso de discos, ocupação de memória, cargas nas máquinas, tráfego de rede e consumo de energia, *logs* de execução do Spark para captura de métricas em nível de aplicação, Yarn e do HDFS, foi montado um *cluster* de monitoração baseado em contêineres [Conceição et al. 2018], composto por 3 servidores físicos Intel Core-i7 2600 3,4 GHz com 16 GB de RAM. Todos os servidores físicos inclusive dos *workers* onde são disparadas as VMs dos executores, foram conectados de forma independente às tomadas do PDU. A potência ativa consumida pelos servidores e ativos foram registrados junto com as métricas da infraestrutura. O ambiente de monitoração contou com ferramentas em versão de software livre do Collectd para interface com o PDU para coletar energia, Sensu para monitorar processador, memória, disco e rede. Para gerenciar filas foi usado o RabbitMQ. O Influxdb foi usado no armazenamento de séries temporais, e o Grafana para visualização dos dados através de telas de acompanhamento (*dashboards*). Estimamos a energia consumida por cada execução como $E = (\sum_{i=1}^{\mathrm{N}} P_i)/N * (t/3600)$, isto é, a energia consumida em watt-hora dada pela potência média de intervalos igualmente espaçados, multiplicada pela duração da aplicação.

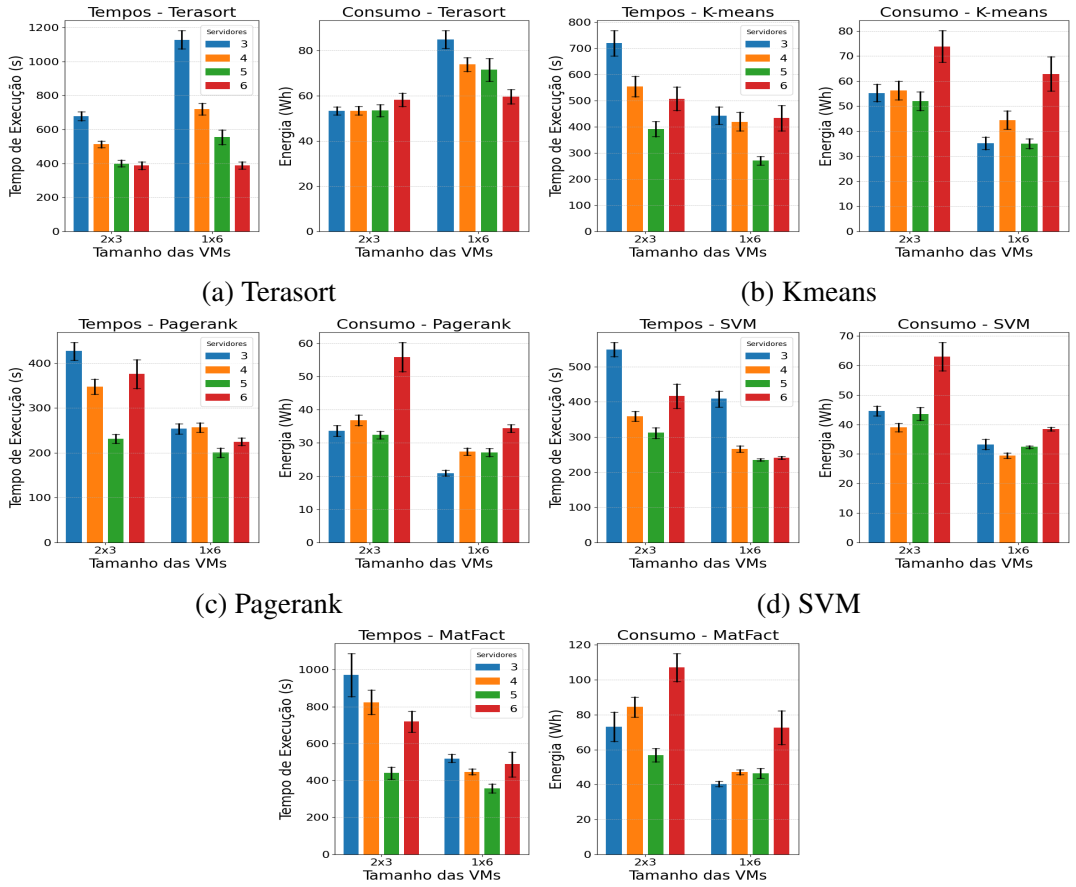
A partir deste ambiente, resumimos a seguir os experimentos realizados neste trabalho:

- **Cargas:** Terasort, K-means, PageRank, SVM, Matrix Factorization
- **Variação 1 (quantidade de servidores):** 3, 4, 5, 6
- **Variação 2 (VMs):** 3 núcleos por VM (2x3), 6 núcleos por VM (1x6)
- **Métricas coletadas:** tempo de execução, desempenho de servidores (CPU, memória, disco, etc.), consumo de energia usando o PDU, métricas de aplicação do Spark

4. Impacto dos servidores e dimensão das VMs no consumo de energia

Os resultados apresentados nesta seção representam um panorama geral das execuções. Os gráficos incluem intervalos de confiança de 95% de 20 execuções para duas medidas:

tempo de execução (segundos) à esquerda e consumo (watt-hora) à direita. Em todas as cargas o esperado é que à medida que se oferece mais recursos, o tempo de execução se reduza. No entanto, conforme será mostrado, nem sempre o consumo de energia sofre uma redução proporcional. Ao acrescentar um novo servidor, um valor P_0 é somado aos demais fatores que compõem o consumo. Todos os fatores que compõem o consumo dependem do tempo decorrido para concluir a aplicação. Por esse motivo, os consumo das cargas se comporta de forma variada.



(a) Terasort (b) Kmeans

(c) Pagerank (d) SVM

(e) MatFact

Figura 2. Tempo de execução e consumo de energia das cargas estudadas.

Terasort (fig. 2a): o aumento do número de servidores reduz o tempo de execução, sendo que o desempenho é melhor com 2 VMs de 3 núcleos. A cada servidor incluído, a potência (não incluída nos gráficos) aumenta de forma similar em ambos os tamanhos de VM. Como o consumo do Terasort é altamente dependente de leituras em disco combinadas com uso de CPU (veja as dimensões de *shuffle* na tab. 1), ter mais VMs contribuiu com a interposição de processamento e entrada/saída, provendo acessos concorrentes ao HDFS e acelerando as execuções. Com relação à variação na quantidade de servidores, observamos que *apesar da aceleração em tempo obtida com mais servidores, nem sempre se obtém uma redução na energia total consumida* – o que nos leva a concluir que a otimização do consumo em aplicações de *big data* não é trivial e muitas vezes não acompanha os ganhos obtidos em tempo de maneira proporcional.

K-means (fig. 2b): O aumento do número de servidores reduz o tempo de execução, somente até um determinado ponto. Ao contrário do Terasort, o desempenho do K-means foi melhor com VMs de 6 núcleos do que com VMs de 3 núcleos. Porém, o

desempenho não melhora consistentemente com o aumento da quantidade de servidores. Por exemplo, há um ponto de mínimo em tempo e consumo observados com 5 servidores em VMs de 6 núcleos. Isso acontece porque o particionamento interno do Spark é incapaz de distribuir bem o trabalho às tarefas no cenário com 4 servidores, problema amenizado ao se adicionar o quinto servidor. De fato, com VMs de 6 núcleos, registramos um uso médio de CPU de 51,4%, 38,4% e 51,4% nas execuções com 3, 4 e 5 servidores respectivamente, sugerindo que *anomalias no particionamento de dados intra-aplicação podem impactar o consumo de energia de forma imprevisível.*

Pagerank (fig. 2c): Em todos os casos o desempenho foi superior em VMs de 6 núcleos. Novamente 5 servidores em VMs de 6 núcleos se mostrou a melhor alternativa combinada entre consumo e tempo de conclusão da tarefa. Apesar do consumo ser similar a 3 servidores, o tempo menor foi obtido em VMs de 6 núcleos com 5 servidores, justamente o momento em que o tempo volta a crescer e o consumo também, por consequência. Pagerank é iterativo e predominantemente intensivo em comunicação, sendo portanto bem menos impactado por disponibilidade de CPU do que o Terasort, por exemplo. Com isso, VMs de 3 núcleos se tornam piores que VMs de 6 núcleos em desempenho. O consumo em VMs de 6 núcleos fica idêntico em 4 e 5 servidores por conta do acréscimo de P_0 ser compensado pelo *speed up*.

SVM (fig. 2d): O comportamento dos tempos de execução foram de relativa semelhança aos do Pagerank. O destaque é que ao oferecer mais recursos para as tarefas, o tempo de execução se reduz inicialmente de maneira bem mais significativa de ao passar de 3 para 4 servidores. Depois o comportamento de redução de tempos é mais suave. A queda abrupta por uma solução em menor tempo reduziu o consumo de maneira tão mais significativa que o acréscimo de consumo P_0, que tornou o menor consumo de energia para o conjunto de 4 servidores com VMs de 6 núcleos. Os consumos para conjuntos de VMs de 3 núcleos acaba por acompanhar os tempos não oferecendo ganhos quando comparados com VMs de 6 núcleos.

Matrix Factorization (fig. 2e): Pode-se observar que novamente as VMs de 6 núcleos desempenharam melhor do que VMs de 3 núcleos. A aceleração do tempo de execução só reduz o consumo total de energia até um determinado ponto, neste caso 5 servidores, e a partir daí se observa acréscimo nos tempos de processamento de modo que o consumo de energia volta a aumentar significativamente. Dentre as cargas estudadas, esta apresentou a maior variabilidade em termos de alocações e durações atípicas de estágios de computação (não ilustrado nos gráficos). Esse comportamento se traduziu em expressiva ociosidade em todas as quantidades de servidores, o que impactou diretamente no consumo de energia.

Conclusões principais: Os resultados apontaram que o *speed up* obtido com mais servidores não se reflete sempre em economia de energia. Além disso, o tamanho das VMs pode influenciar no consumo porque impacta diretamente em como o *framework* de processamento paralelo (no caso, Spark) percebe o nível de paralelismo disponível. A diversidade de tendências observadas nas cargas estudadas levou a questionar quais fatores seriam capazes de explicar tais comportamentos. Na seção seguinte, descemos um nível em nossa análise e buscamos estudar o consumo de energia frente às métricas de utilização de recurso (CPU, memória, disco e rede) capturadas pela monitoração.

5. Importância de fatores de desempenho no consumo de energia

Visando determinar como as métricas coletadas pela monitoração impactam nos resultados obtidos, foram realizadas regressões multifatoriais para cada carga com todos os fatores coletados pela monitoração. Foram considerados: o percentual médio de uso dos processadores disponibilizados durante as execuções (CPU-AVG), o somatório dos dados escritos (D-WRITE) e lidos em disco (D-READ) em GB, o somatório dos dados transmitidos na rede durante a execução (NET) em GB e o impacto em se utilizar 2 VMs de 3 núcleos ou uma VM de 6 núcleos (VM-CONFIG), sendo a primeira opção a referência. Estes fatores assumiram como variável dependente o consumo de energia em watt-hora e todos os coeficientes são apresentados com um intervalo de confiança de 95%.

Tabela 2. Modelo e resultados das regressões lineares em cada carga.

$$\text{ConsumoEnergia} = \beta_0 + \beta_1 * \text{CPU-AVG} + \beta_2 * \text{D-READ} + \beta_3 * \text{D-WRITE} + \beta_4 * \text{NET} + \beta_5 * \text{VM-CONFIG}$$

		β_0 (Intercepto)	β_1(%)	β_2(GB)	β_3(GB)	β_4(GB)	β_5(0 ou 1)
Terasort	Coef.	87.2 ± 3.6	-0.8 ± 0.1	8.4 ± 1.7	-330.0 ± 76.0	0.0013 ± 0.0003	26.4 ± 5.1
$R^2 = 0.89$	p	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$
Pagerank	Coef.	27.2 ± 7.5	-0.4 ± 0.0	-15.2 ± 11.7	218.2 ± 81.7	-0.0006 ± 0.0002	31.5 ± 7.1
$R^2 = 0.88$	p	$< .001$	$< .001$	0.012	$< .001$	$< .001$	$< .001$
Kmeans	Coef.	47.7 ± 4.9	-0.9 ± 0.1	-17.1 ± 3.7	-24.0 ± 46.6	0.0009 ± 0.0002	65.3 ± 4.7
$R^2 = 0.94$	p	$< .001$	$< .001$	$< .001$	0.31	$< .001$	$< .001$
SVM	Coef.	54.2 ± 2.8	-0.4 ± 0.1	-46.8 ± 12.4	888.5 ± 190.7	-0.0024 ± 0.0005	15.0 ± 5.1
$R^2 = 0.95$	p	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$
MatFact	Coef.	67.7 ± 6.1	-1.3 ± 0.1	18.8 ± 9.6	-66.8 ± 89.4	-0.0001 ± 0.0002	79.9 ± 7.9
$R^2 = 0.94$	p	$< .001$	$< .001$	$< .001$	0.142	0.212	$< .001$

A tabela 2 mostra que o menor coeficiente de determinação R^2 é 0,88 ou seja os modelos obtidos das regressões representam pelo menos 88% da variabilidade dos dados chegando 95% para o SVM. Os valores p em sua grande maioria atendem aos requisitos de hipótese não nula (isto é, temos evidências estatísticas para acreditar que os coeficientes são significativos e diferentes de zero). O Terasort apresentou o maior intercepto, sugerindo que, independentemente das variáveis explicativas, seu valor base é maior – isso fica evidente ao observar que o Terasort é a carga que mais consome energia dentre as consideradas. Ao constatar altos valores de R^2 para todas as cargas, podemos concluir que o fenômeno do consumo de energia pode ser bem explicado (mas não de forma trivial) a partir das métricas de desempenho coletadas pela monitoração.

Em seguida, avaliamos o impacto incremental de fatores no modelo através de uma regressão linear hierárquica. Partimos de uma regressão com o fator CPU-AVG e adicionamos os fatores D-READ, D-WRITE, NET, e VM-CONF de forma a observar os incrementos no R^2. Os resultados estão apresentados no mapa de calor da figura 3. O fator CPU-AVG sozinho apresenta valores de R^2 muito baixos para todas as aplicações, indicando que a necessidade de outros fatores para que o consumo seja melhor explicado. Dentre todas as cargas, o Terasort é o que mais pode ser explicado através de CPU-AVG apenas – por ser uma aplicação com poucos estágios que demandam muito do disco e também da CPU para ordenar os itens em memória, grande parte do consumo de energia neste caso vem do uso intensivo desse recurso. Adicionar D-READ e D-WRITE melhora substancialmente R^2 para quase todas as aplicações. Isso se dá por conta dos modelos

passarem a considerar a dimensão de entrada e saída das aplicações. Já o K-means apresenta pouca melhora em R^2, só ficando melhor modelado com a inclusão de NET, já que o desbalanceamento de pontos em relação aos respectivos centroides é expressivo nas entradas do SparkBench. Por fim, vemos um incremento de R^2 menor ao incluir VM-CONFIG na maioria das cargas, indicando que os impactos causados pela escolha do tamanho da VM já estão bem capturados a partir dos outros fatores já incluídos.

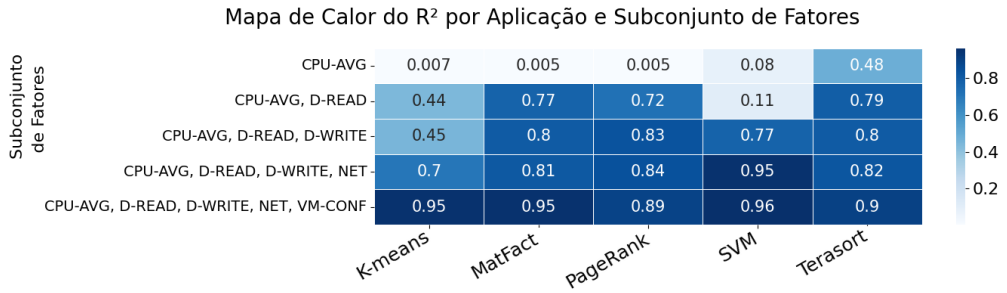


Figura 3. Regressão linear hierárquica: fatores importantes a cada carga.

Conclusões principais: O uso de CPU e uso de disco modelam o consumo de energia para quase todos os casos, sendo que o K-means só atinge modelagem razoável com a inclusão do fator dados trafegados em rede. Já o tamanho das VMs teve impacto significativo para os algoritmos K-means, Matrix Factorization e Terasort. As regressões completas explicam bem o consumo de energia, de modo que possuir sistemas com monitoração integradas ao ambiente, geram indicadores que facilitam entender onde se pode economizar energia processando *big data*. Muitas vezes, com apenas alguns fatores mesmo que isolados, dá para se modelar bem o consumo de energia. Mas, o mais importante a destacar é: *aplicações Big Data possuem comportamentos tão particulares que exigem uma análise transversal do consumo de energia que inclua todos os recursos utilizados, e não apenas como um produto da utilização de CPU*.

6. Impacto da utilização de recursos no nível de aplicação

Ainda que se compreenda os fatores mais importantes para o consumo de energia de uma carga, é importante também identificar características próprias da execução que levaram ao comportamento observado. Uma metodologia multicamadas de ponta a ponta como esta *permite ao usuário do sistema ajustar de maneira sistemática suas aplicações de modo a economizar energia*. Esta parte do trabalho avaliou o impacto da alocação de tarefas pelo Spark sobre o consumo de energia. Em um servidor com a capacidade de executar T_{max} tarefas Spark em paralelo, o potencial máximo de alocação de uma execução que durou D segundos seria dado por $T_{max} * D$. Com isso, definimos *ScoreUtil* (entre 0 e 1) de um servidor como sendo a proporção desse potencial de fato preenchida com tarefas Spark.

A tabela 3 apresenta o *ScoreUtil* dos servidores para as melhores configurações de VM observadas em cada carga. Quando contrastada essa medida com os consumos de energia reportados na figura 2, percebe-se que um maior consumo de energia geralmente está associado a um preenchimento sub-ótimo do recurso disponível com tarefas (menor *ScoreUtil*). Nesse caso, é possível identificar a partir de certo ponto as cargas que não se beneficiam de mais servidores, que ligados, somam em consumo desnecessário de energia. Além disso, o desbalanceamento na utilização de recursos pelos servidores desempenha um papel importante. Tomando como exemplo MatFact em 6 servidores, vê-se

Tabela 3. *ScoreUtil:* utilização de recurso à medida que se aumenta a quantidade de servidores, para a melhor configuração de VM identificada para cada carga.

	VM	3 servidores			4 servidores			5 servidores			6 servidores		
		min.	méd.	max.	min.	méd.	max.	min.	méd.	max.	min.	méd.	max.
Terasort	2x3	0,86	0,87	0,88	0,77	0,80	0,86	0,68	0,72	0,79	0,63	0,67	0,78
Kmeans	1x6	0,36	0,55	0,69	0,49	0,58	0,69	0,34	0,47	0,70	0,38	0,44	0,51
Pagerank	1x6	0,53	0,62	0,79	0,36	0,47	0,65	0,40	0,48	0,56	0,28	0,35	0,63
SVM	1x6	0,67	0,77	0,83	0,68	0,71	0,76	0,60	0,68	0,74	0,57	0,63	0,70
MatFact	1x6	0,40	0,56	0,87	0,30	0,45	0,86	0,29	0,42	0,82	0,13	0,28	0,81

que existem servidores muito bem utilizados ($ScoreUtil = 0,81$), enquanto a maioria dos outros servidores continuam praticamente ociosos (Min. $ScoreUtil = 0,13$ e Média $ScoreUtil = 0,28$). Pagerank e Kmeans manifestam um progressivo desbalanceamento na utilização, similar ao MatFact. Em outro outro extremo, observa-se que Terasort e SVM tiveram os menores graus de desbalanceamento em *ScoreUtil*. De fato, as cargas com processamento mais regular e denso (Terasort e SVM) permitiram uma melhor utilização quando comparadas a processamento mais esparso e irregular, típico de cargas que processam grafos e dados espaciais como Pagerank e Kmeans.

Conclusões principais: É importante uma abordagem de monitoração multicamadas que integre e associe todas essas fontes de dados de monitoração. Além disso, descer no nível da aplicação é essencial para economizar energia, já que diversos fatores como balanceamento de carga, comunicação e utilização de recursos só podem ser ajustados a partir de ações nesse nível. *Fica evidente que estratégias de economia de energia em Big Data podem se beneficiar de uma caracterização caso-a-caso das cargas.*

7. Conclusões

Este trabalho avaliou o desempenho e o consumo de energia de cargas *big data* em um ambiente de servidores em nuvem. O ambiente de monitoração contou com um PDU, que permitiu coletar com alta precisão o consumo de energia dos experimentos. Os resultados mostraram que não há soluções triviais para se economizar energia em cargas *big data*. Definir os fatores de maior impacto no consumo de cada carga é possível por meio de regressões multifatoriais, até mesmo com poucos fatores. Uma modelagem hierárquica se mostrou efetiva e de relativa simplicidade para isso. Escolher o tamanho das VMs também foi significativo, tanto para o desempenho como para o consumo. Há casos em que VMs menores favorecem o paralelismo das tarefas em um nível diferente da camada das aplicações. Adicionar recursos na expectativa de que o *speed up* economizaria energia nem sempre foi suficiente. A prática mostrou que tipicamente há um ponto de custo-beneficio que combina tempo e consumo de energia, podendo ser obtido experimentalmente. Foi fundamental estabelecer uma abordagem multicamadas que integrou e associou todas as fontes de dados de monitoração combinada com informações da aplicação para apontar como economizar energia. Afinal, vários fatores tais como balanceamento de carga, operações de entrada e saída e utilização de recursos só podem ser ajustados a partir de ações nesse nível.

8. Agradecimentos

Este trabalho foi parcialmente fomentado pela Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-

14), H2020-EUBR-2015 EUBra-BIGSEA, H2020-EUBR-2017 Atmosphere e FAPEMIG Universal APQ-00202-24.

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.

Anand, V., Xie, Z., Stolet, M., De Viti, R., Davidson, T., Karimipour, R., Alzayat, S., and Mace, J. (2022). The odd one out: Energy is not like other metrics. In *HotCarbon 2022: 1st Workshop on Sustainable Computer Systems Design and Implementation*.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79-80:3 – 15. Special Issue on Scalable Systems for Big Data Management and Analytics.

Asyabi, E., Sharifi, M., and Bestavros, A. (2018). ppxen: A hypervisor cpu scheduler for mitigating performance variability in virtualized clouds. *Future Generation Computer Systems*, 83:75–84.

Baker, T., Al-Dawsari, B., Tawfik, H., Reid, D., and Ngoko, Y. (2015). Greedi: An energy efficient routing algorithm for big data on cloud. *Ad Hoc Networks*, 35:83–96.

Bernardo, F., Ferro, M., Vieira, V., Silva, G., and Schulze, B. (2020). Em busca de uma inteligência artificial ecologicamente viável: Um estudo de caso do consumo energético de algoritmos de íarvore de decisão. In *Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 179–190. SBC.

Bernardo, F., Yokoyama, A., Schulze, B., and Ferro, M. (2021). Avaliação do consumo de energia para o treinamento de aprendizado de máquina utilizando single-board computers baseadas em arm. In *Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 60–71. SBC.

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)*.

Forte, C. H., Manacero, A., Lobato, R. S., and Spolon, R. (2018). An energy-aware task scheduler based in ownership fairness applied to federated grids. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00030–00033. IEEE.

Gonçalves, T. d. S., Beck, A. C. S., and Lorenzon, A. F. (2023). Explorando a variabilidade de processo para otimizar a eficiência energética em servidores de nuvem. In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 229–240. SBC.

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.

Higgs, E. (2018). ehiggs/spark-terasort. `https://github.com/ehiggs/spark-terasort`.

IEA (2022). Data centres and data transmission networks. https://www.iea.org/reports/data-centres-and-data-transmission-networks. Acessado em Maio de 2023.

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.

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

Li, H., Wang, H., Fang, S., Zou, Y., and Tian, W. (2020). An energy-aware scheduling algorithm for big data applications in spark. *Cluster Computing*, 23:593–609.

Li, M., Tan, J., Wang, Y., Zhang, L., and Salapura, V. (2015). Sparkbench: a comprehensive benchmarking suite for in memory data analytic platform spark. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 53. ACM.

Li, M., Tan, J., Wang, Y., Zhang, L., and Salapura, V. (2017). Sparkbench: a spark benchmarking suite characterizing large-scale in-memory data analytics. *Cluster Computing*, 20(3):2575–2589.

Mashayekhy, L., Nejad, M. M., Grosu, D., Zhang, Q., and Shi, W. (2015). Energy-aware scheduling of mapreduce jobs for big data applications. *IEEE Trans. Parallel Distrib. Syst.*, 26(10):2720–2733.

Ousterhout, J. (2018). Always measure one level deeper. *Communications of the ACM*, 61(7):74–83.

Pesce, M. (2021). Cloud computing's coming energy crisis-the cloud's electricity needs are growing unsustainably. *IEEE Spectr*.

Saeed, M. M., Al Aghbari, Z., and Alsharidah, M. (2020). Big data clustering techniques based on spark: a literature review. *PeerJ Computer Science*, 6:e321.

Spark, A. (2015). Tuning spark. https://spark.apache.org/docs/2.2.0/tuning.html. Acessado em Agosto de 2019.

Volpini, N. D. O., Conceição, V. S., Pontes, R. L., and Guedes, D. (2018). Uma análise do consumo de energia de ambientes de processamento de dados massivos em nuvem. In *Anais do XVII Workshop em Desempenho de Sistemas Computacionais e de Comunicaç ao, Natal-RN. Sociedade Brasileira de Computaçao (SBC)*.

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.