Capítulo 5

Explorando Técnicas de Compressão para Melhorar a Eficiência de Tratamento de Dados IoT

Alexandre Luis de Andrade, Rodrigo da Rosa Righi
Universidade do Vale do Rio dos Sinos

Resumo

Esse capítulo de livro descreve o papel, por vias esquecido, que a compressão de dados pode ter no âmbito de processamento de alto desempenho. Hoje, com a concretização do uso de Internet das Coisas e implantação onipresente de algoritmos de Inteligência Artificial, o tratamento de dados ganham muita ênfase para que consigamos desempenho. Isso porque ainda seguimos com a sentença que eles são manipulados numa ordem de grandeza temporal bem maior que processamentos feitos pela CPU. Diante disso, o capítulo de livro apresenta os cenários que podem usufruir de aspectos de compressão de dados, bem como discute diferentes técnicas, sub-tempos envolvidos e casos de uso. Acreditamos que compressão possa ser muito explorada nos novos cenários de transformação digital, podendo também ser protagonista para a modelagem de sistemas computacionais de alto desempenho.

5.1. Introdução

O processo de transformação digital tem impulsionado a utilização de tecnologias digitais nas mais diversas áreas. Em especial, a adoção de dispositivos e sensores conectados, referidos como Internet das Coisas, ou *Internet of Things* (IoT), vem trazendo novas oportunidades e promovendo o avanço da Industria 4.0 (I4.0), na medida em que possibilita redução de custos, maior eficiência e incremento de qualidade (Marinagi et al. 2023) (Aheleroff et al. 2020). Cabe também destacar o impacto que a COVID-19 trouxe na aceleração dos processos de transformação digital e no fomento da disrupção em vários setores, tais com impulso na adoção da telemedicina (Castiglione et al. 2021) e, em particular, o uso de dispositivos vestíveis, em inglês *wearables*.

Esse cenário, com as companhias investindo cada vez mais em processos de automação e infraestruturas escaláveis, leva a um aumento exponencial de sensores conectados. Além desse fator, a popularização dos dispositivos *wearables* também contribui

para geração de uma grande volumetria de dados. Segundo a International Data Corporation (IDC), em 2025 estima-se 41,6 bilhões de dispositivos IoT conectados, gerando mais de 79 Zettabytes (ZB) de dados (Lu et al. 2020). Contudo, ao lidar com tal volumetria nos deparamos com dois principais desafios, que são a transmissão e o armazenamento destes dados.

Quanto à transmissão, devemos considerar que os dispositivos IoT utilizam basicamente redes sem fio, via radio frequência, para conexão e transmissão de dados na rede, e caso a rede não esteja adequadamente dimensionada, pode resultar em perda de pacotes e necessidade de retransmissão. Para sistemas que são sensíveis ao tempo e à qualidade da informação, tais como *healthcare*, para monitoramento de sinais vitais, a transmissão dos dados representa um fator crítico para um diagnóstico preciso e eficiência no tempo de reação (Idrees and Khlief 2023).

Ao tratar do armazenamento desses dados, vale considerar um dos principais elementos da transformação digital que é a computação em nuvem. A computação em nuvem foi um fator chave para a aceleração da transformação pois trouxe mais agilidade, facilidade de alocação e escalabilidade, atendendo rapidamente às necessidades do negócio (Dwivedi et al. 2022). Nesse contexto, o armazenamento de dados nuvem constitui um grande avanço, devido aos seus vastos recursos de armazenamento e escalabilidade, oferecendo segurança de dados, facilidade de acesso e recuperação em caso de desastres. Contudo existe um custo financeiro associado que precisa ser considerado, demandando estratégias adequadas de armazenamento e recuperação da informação (Aggarwal and Lan 2020).

Desse modo, para impulsionar áreas como a nova *healthcare* e a I4.0, que fazem uso extensivo de sensores e geram grande volume de dados, é necessário encontrar métodos para mitigar tais fatores que podem degradar o desempenho e a qualidade do resultado final. Nesse sentido, surgem algumas abordagens como alternativas para lidar com esses problemas, tais como elasticidade computacional, balanceamento de carga, computação de alto desempenho, ou *High-Performance Computing* (HPC), agregação de mensagens, e compressão de dados que vem ganhando espaço. De fato, a técnicas de compressão de dados não constitui algo novo, contudo ressurge nesse novo contexto como alternativa eficiente e de baixa complexidade de implantação (Gia et al. 2019) (Khlief and Idrees 2022) (Das and Rahman 2021).

Para ilustrar esse contexto, na Figura 5.1 estão apresentados três cenários possíveis de transmissão e armazenamento de dados, sendo em (1) sem aplicação de técnicas de compressão, (2) com compressão de dados e (3) comprimindo e aplicando descompressão antes de armazenar. Nesses cenários estão destacados os tempos envolvidos nas diferentes etapas, sendo o tempo necessário para transmissão (T_t), para compressão (T_c) e o tempo para descompressão de dados (T_d). Com isso, os tempos totais envolvidos nos cenários apresentados são $T_1 = T_t$, $T_2 = T_c + T_t$, e $T_3 = T_c + T_t + T_d$. De modo que, se dimensionado adequadamente, o esperado é $T_3 > T_2 > T_1$.

Dado esse contexto, o objetivo desse trabalho é abordar diferentes técnicas de compressão de dados e de que modo podem contribuir na melhoria de desempenho no processos de compactação e restauração da informação, levando a uma maior eficiência na sua transmissão e armazenamento de dados. O estudo também explora cenários práticos

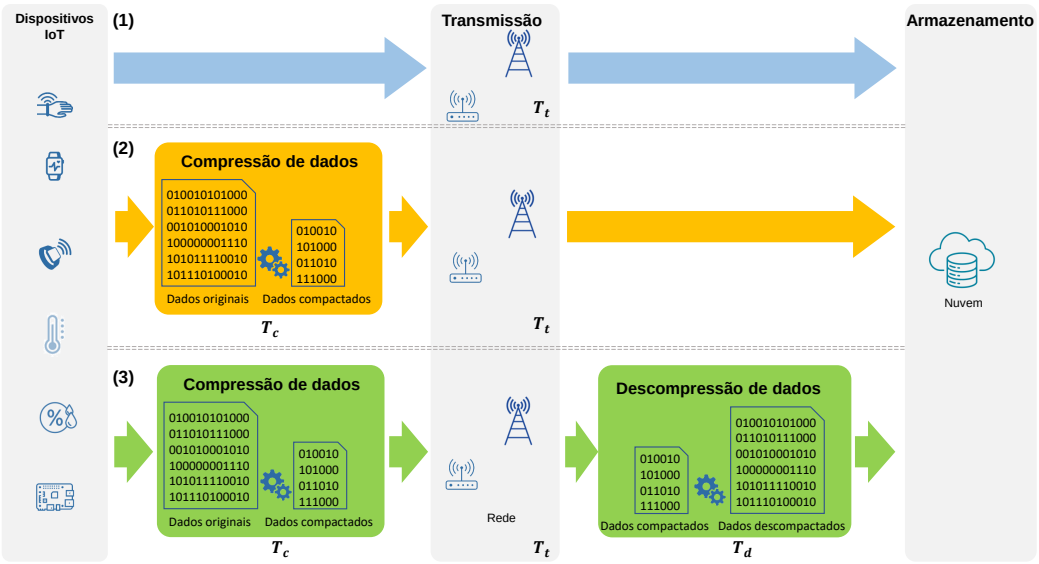

Figura 5.1. Cenários possíveis de transmissão e armazenamento da informação, onde (1) não aplica técnica de compressão; em (2) os dados são compactados e então transmitidos para serem armazenados e em (3) os dados são comprimidos e então transmitidos, contudo, são descompactados antes de serem armazenados.

de monitoramento de sistemas de healthcare e I4.0, nos quais as técnicas de compressão, mesmo que já extensamente estudadas, podem ser adaptadas e combinadas para levar a soluções inovadoras.

O restante do texto está organizado com a Seção 5.2 que aborda o contexto atual de geração de dados e seus desafios, seguido pela Seção 5.3 que analisa o potencial de uso dos métodos de compressão. Na Seção 5.4 serão analisados os diferentes tipos de algoritmos e principais aplicações, enquanto que a Seção 5.5 discute a combinação de diferentes técnicas para atender demandas IoT relacionadas com edge, fog e cloud comouting. Na sequência, tem-se a Seção 5.6, a qual traz as principais métricas adotadas para avaliar desempenho e a qualidade dos algoritmos de compressão. Por fim, na Seção 5.7 será feita uma análise conclusiva do tema de compressão de dados no cenário atual de computação de alto desempenho e alta demanda de dados.

5.2. Demanda de dados IoT e desafios

A presença de dispositivos conectados vem crescendo em um ritmo exponencial nos último anos, cada vez menores e com custos acessíveis, estão revolucionando diversas áreas. A área da saúde, por exemplo, com pacientes assistidos por IoT podem ser supervisionados ininterruptamente através dispositivos vestíveis, permitindo assim que situações de risco sejam detectadas e tratadas em tempo (Fischer et al. 2020). Ainda na área da saúde, o advento do COVID-19 impulsionou o *ehealth* com uso extensivo de dispositivos e trouxe a regulamentação da telemedicina, com sistemas remotos de monitoração.

Do mesmo modo, a I4.0 apoiada na integração com dispositivos IoT em suas instalações e, somada à computação e análise em nuvem, está revolucionando a forma

como as empresas fabricam, melhoram e distribuem seus produtos. A operação de fabricação inteligente, atuando de maneira cruzada com tecnologias automatizadas de aquisição e simulação de dados em tempo real, permite a identificação instantânea de problemas físicos na produção, com agilidade de reação para ações corretivas e otimização do desempenho em todo o sistema de fabricação, conforme Figura5.2 (Elkaseer et al. 2018).

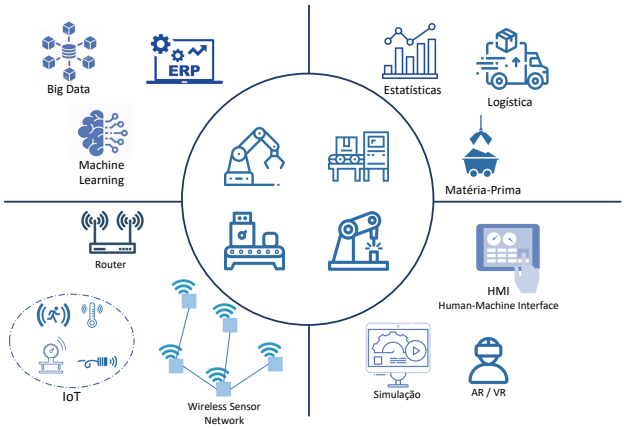

Figura 5.2. Framework de Intustria 4.0, representando a relação cruzada entre tecnologias, integrando diferentes áreas de manufatura, operação e logística.

Posto esse cenário, fica evidente a necessidade crescente de transferência de dados através da rede, seja para monitorar, executar comandos remotos ou mesmo disponibilizar informação em bases para análise e tomada de decisão. Entretanto, para que esses processos atendam às necessidades propostas, alguns fatores críticos de transmissão de dados precisam ser considerados, pois podem influenciar diretamente na qualidade, na eficiência e no desempenho do sistema. Os principais fatores mais citados na literatura são a i) ocupação da largura de banda, ii) taxa de transferência de dados, do inglês *throughput* e iii) latência de rede.

A largura de banda corresponde à quantidade máxima de dados que podem ser transmitidos em um determinado período de tempo em uma conexão de rede, sendo medida em bits por segundo. De modo que, se a transferência de dados excede a capacidade ocorre atraso na transmissão. O *Throughput* dos dados na rede, se refere à taxa em que os dados são transmitidos com êxito de um local para outro em um determinado período. E a latência de rede é o atraso na transmissão de dados da origem ao destino. Embora esses três elementos sejam diferentes, eles estão diretamente relacionados entre si: baixa largura de banda pode resultar em maior latência e menor *throughput*.

Como já citado, se esses fatores estiverem mal dimensionados, podem prejudicar a qualidade do serviço e, portanto, sua confiabilidade e limitar a escalabilidade do sistema. De fato, isso contrasta com o surgimento de novos aplicativos para sistemas e processos críticos, os quais vem se exigindo requisitos de confiabilidade e latência cada vez mais rígidos, conforme Figura 5.3. Na I4.0, em particular, o controle de robôs de alta precisão e, na área automotiva, a operação de veículos autônomos precisam oferecer confiabilidade superior à ordem de cinco noves (99,999%) e latência de milissegundos (Park et al. 2020). A automação na industria com base em comunicação *wireless* deve garantir confiabilidade

na ordem de sete noves (99,99999%) e latência abaixo de 1 ms, semelhante aos padrões de sistema de tempo real baseado em Ethernet (Berardinelli et al. 2018).

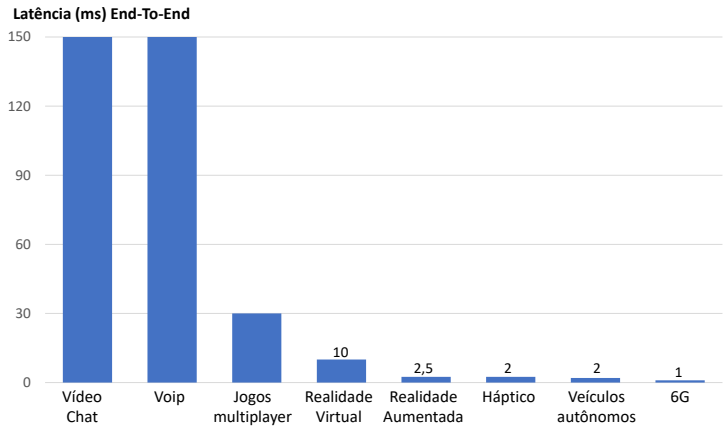

Figura 5.3. Latência esperada por tipo de aplicação (Panwar 2020).

Da mesma forma, em sistemas para processamento de mídia, a latência tem impacto direto sobre a qualidade e a escalabilidade (Abuin et al. 2022). Ou seja, melhorar a qualidade implica em aumento da resolução e aumento do número de quadros por segundo, tornando maior a quantidade de dados transmitidos. Por outro lado, aumentar a escalabilidade, significa levar a mídia para um número maior de usuários e aumentar o número de canais.

Nesse sentido, se procura estabelecer o equilíbrio para o triângulo formado por latência, qualidade e escalabilidade, enquanto se apresentam de novos algoritmos e novos modos de compressão de arquivos (Chen et al. 2020).

5.3. Potencial de compressão de dados

A compressão de dados não é uma ciência nova. Logicamente, existem novas técnicas e soluções aplicadas a tipos de arquivos modernos, mas em geral, as soluções existentes no mercado datam de mais de duas décadas. Apesar de não ser uma novidade, entendemos que compressão de dados ganha relevante importância no cenário atual, onde as áreas de Inteligência Artificial, Big Data, Cidades Inteligentes e IoT ganham destaque. Em particular, vale ressaltar que tais demandas geram muitos dados, os quais muitas vezes precisam ser armazenados, ou ainda transferidos entre uma máquina e outra pela rede de interconexão.

Diante disso, a ideia de compressão de dados é bem simples: diminuir o tamanho de um arquivo (o qual é composto por um fluxo de dados ou mesmo um arquivo tradicional no sistema de arquivos do sistema operacional). Tal diminuição fará, portanto, que se ocupe menos espaço em disco para armazenamento e se ocupe menos largura de banda para a sua transferência pela rede. Apesar de claros os benefícios da técnica, vale atentarmos para uma série de questões. Para explicá-las, vamos usar as notações abaixo:

- $tempo\_comunicacao(dados)$ - Função que torna o tempo para a transferência de

uma região de *dados*. Perceba que *dados* pode estar na sua forma original ou na compra comprimida. Por questões de simplicidade, não estamos levando em conta a tecnologia de rede empregada, tampouco os protocolos usados na comunicação.

- $tempo\_armazenamento(dados)$ - Função que retorna o tempo de armazenamento em disco para uma quantidade de dados denotada por *dados*. Por questão de simplicidade, não estamos considerando a tecnologia de disco rígido empregada.
- $temp\_compressaao(dados, tecnica)$ - Função que apresenta o tempo para aplicar o motor de compressão usando a estratégia *tecnica* sobre uma região de *dados*.
- $tempo\_descompressaao(dados, tecnica)$ - Procedimento que retorna o tempo para aplicar o motor de descompressão usando a estratégia *tecnica* sobre uma região de *dados*.

Com as colocações acima, é possível denotar ao menos dois cenários para explorar o potencial de compressão de dados: (i) cenário de somente transferência de dados; (ii) cenário de transferência e armazenamento. No cenário (i), a ideia é que a compressão seja pertinente para reduzir o tempo de comunicação. Ela é particularmente interessante em redes de larga cobertura ou aquelas que tendem a possuir alto nível de ruido, como algumas transferências por rádio frequência. A Inequação 5.3 apresenta o objetivo final do cenário (i). Nela, *dados* denotam o conjunto de dados original, enquanto $dados'$ representa o resultado após a aplicação de uma determinada *tecnica* de compressão. No cenário (i), temos que a semântica de transferência é exatamente a mesma, ou seja, partimos de dados que devem ser transferidos e chegamos até a transferência deles por completo para a máquina destino. Logicamente, para a compressão há as etapas de comprimir, transferir o material resultante e descomprimi-lo adequadamente.

$$tempo\_compressao(dados, tecnica) + tempo\_comunicacao(dados') + tempo\_descompressao(dados') < tempo\_comunicacao(dados) \quad (1)$$

No cenário (ii), o foco é armazenar os dados no destino na forma comprimida. Esse cenário aparece usualmente quando os dados precisam ser transferidos e agregados em um servidor para posterior análise, a qual não acontece de imediato após a transferência. A Inequação 2 apresenta a estratégia para que valha a pena o uso de compressão nesse cenário. Como é possível perceber, aplica-se a compressão e trabalha-se com a transferência e armazenamento dos dados comprimidos. Essa inequação não leva em consideração a redução do espaço em disco que pode ocorrer no armazenamento da forma comprimida, e sim somente o aspecto temporal do uso da técnica de compressão. Após o armazenamento dos dados na máquina destino, o uso deles pode ou não requisitar a descompressão. Se estivermos trabalhando com compressão do tipo *lossy*, os próprios dados transmitidos já estão no padrão que podem ser trabalhados e manipulados, não requisitando descompressão. Em contra-partida, a transferência e o armazenamento de arquivos que denotam documentos pode ocorrer na forma comprimida, mas o uso necessária

vai incorrer numa descompressão. Essa descompressão pode ser realizada somente em memória ou em espaço de armazenamento temporário, bem como ser realizada de forma assíncrona sem interferir nas atividades críticas de tempo para o usuário final.

$$tempo\_compressao(dados, tecnica) + tempo\_comunicacao(dados') + tempo\_armazenamento(dados') < tempo\_comunicacao(dados) + tempo\_armazenamento(dados) \quad (2)$$

Tanto para o cenário (i) quanto o cenário (ii), a noção de grão é importante para que seja possível entender as inequações apresentadas. Uma vez que estamos analisando o tempo de transferência pela rede de computadores, tempo de armazenamento em memória secundária e os tempos dos motores de compressão e descompressão, o formato e o tamanho dos dados é crucial. Por exemplo, na transferência de 100 bytes, possivelmente tenhamos uma situação onde o uso da técnica de compressão pode ser proibitivo, uma vez que a própria chamada de compressão e descompressão acarretariam em sobrecarga a qual não seria sobrepassada pelos ganhos de comunicação na transferência da forma comprimida. Entretanto, a manipulação de arquivos de 1 Mbyte pode ser profícua com a compressão, desde que o ganho com a transmissão exceda a sobrecarga das traduções de original para comprimido e vice-versa. Aqui vale ressaltar que as CPUs nas máquinas origem e destino desempenho um papel crucial, uma vez que contribuem para a velocidade das execuções dos algoritmos de compressão e descompressão de dados. Por fim, o tipo de dados de um arquivo é importante para denotar o potencial de compressão. Usando o algoritmo empregado no software WinZip, enquanto que em arquivos texto ASCII é possível atingir taxas de compressão de até 73%, bibliotecas DLL ou arqivos EXE possuem taxas de compressão de até 50% (em média)[1].

5.4. Tipos de compressão de dados

Os algoritmos de compressão de dados são tradicionalmente classificados em dois tipos: com perdas (ou não inversíveis), do inglês *lossy*, ou sem perdas (ou inversíveis), do inglês *lossless*. Embora o primeiro algoritmo geralmente forneça um resultado mais significativo no que se refere à taxa de compressão, esse algoritmo não possibilita reconstruir totalmente os dados originais. Por outro lado, os sistemas de compressão sem perdas reduzem o tamanho dos dados transmitidos ou armazenados, focando em reduzir a redundância de informações da origem, preservando a entropia da informação e permitindo a reconstrução integral dos dados originais. Comparativamente, a taxa de compressão dos algoritmos *lossless* são menores que as do algoritmo *lossy*.

Os algoritmos *lossless* comprimem o arquivo geralmente por codificação de repetição, basicamente substituindo uma palavra ou caractere com maior frequência por um código que utiliza menos espaço de armazenamento. Esses algoritmos comumente utilizam estratégias de codificação de redundância mínima ou método de dicionário. A técnica de redundância mínima consiste em representar os caracteres com maior número de ocorrências utilizando menos bits. Um exemplo de algoritmo muito conhecido que

[1]Detalhes em: https://kb.corel.com/en/125890

aplica essa técnica é Codificação de Huffman. Quanto ao método do dicionário, é aplicado um arquivo como sendo um dicionário de expressões com os seus valores correspondentes em bits, como exemplo de algoritmo que aplica essa técnica temos Lempel-Ziv-Welch (LZW). Outros algoritmos *lossless* amplamente conhecidos são ZIP e RAR, e tem maior rendimento, como citado na seção anterior, quando aplicados em arquivos de texto ASCII.

Os algoritmos *lossy*, por outro lado, elimina uma certa quantidade de dados que não é perceptível e, desse modo, não permite que a informação seja restaurada em sua forma original, contudo reduz significativamente o tamanho, conforme ilustrado no Figura 5.4. O esperado deste tipo de algoritmo é que permita uma análise similar entre os dados comprimidos e os originais sem eliminar informações relevantes e com a mais alta taxa de compressão possível. Ou seja, essa técnica é benéfica se a qualidade dos dados não for prioridade mas adequada para enviar ou armazenar os dados. Esse tipo de compactação de dados é usado principalmente para comprimir dados de multimídia como voz, vídeo e imagens.

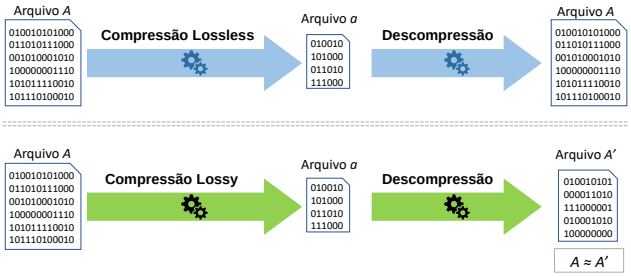

Figura 5.4. Dois principais métodos de compressão, Lossless e Lossy. O método Lossy proporciona uma maior taxa de compressão, contudo os dados restaurados a partir da descompressão não correspondem aos do arquivo original.

Nessa categoria do algoritmo de compressão com perdas, os métodos ”lineares por partes”, em inglês *piecewise linear*, são frequentemente utilizados na indústria, pois demandam baixo poder processamento e podem ser aplicados em tempo real no momento da aquisição dos dados (Roy and Nikolaidis 2022). Estes métodos consideram que o sinal segue uma linha reta enquanto os pontos amostrados estivem contidos dentro de uma tolerância especificada. Os dados considerados redundantes ou mesmo próximos de uma certa tendência linear são descartados e, desse modo, tais dados poderiam ser recuperados futuramente com baixo erro, utilizando técnicas de interpolação. Dentre os algoritmos *lossy* que utilizam esse método que são adotados na I4.0 se destacam o Boxcar/Backslope e o Swinging Door Trending (Khan et al. 2020).

A união de dois algoritmos, Boxcar e BackSlope, resulta numa técnica em que se armazena apenas os pontos que satisfazem as condições de ambos. Os parâmetros do algoritmo são a largura da janela do Boxcar e a largura da janela do Backslope. A partir da largura de janela informada, o algoritmo estabelece um limite de variação horizontal em relação ao último valor armazenado (BoxCar). Conforme apresentado na Figura 5.5a, a cada novo valor reportado são calculados limites de variação diagonal, utilizando a mesma largura de janela (BackSlope). Desse modo, assim que um valor violar duas áreas estabelecidas pelas linhas BoxCar e BackSlope, o valor deve ser armazenado.

O algoritmo Swinging Door Trending, apresentado na Figura 5.5b, utiliza como parâmetros o tempo máximo de compressão e desvio de compressão. Sendo que quanto maior o tempo máximo de compressão e o desvio de compressão, maior será a taxa de compressão. Uma área de cobertura á criada no formato de um paralelogramo com altura igual a duas vezes o desvio de compressão desde o último valor armazenado até o último valor recebido. Se algum dos pontos coletados entre o último valor armazenado e o valor atual estiver fora da área, todos os pontos são descartados, com exceção do penúltimo ponto recebido, que é armazenado. Ainda, sempre que o tempo entre o último valor armazenado for superior ao tempo máximo de compressão último valor deve ser armazenado.

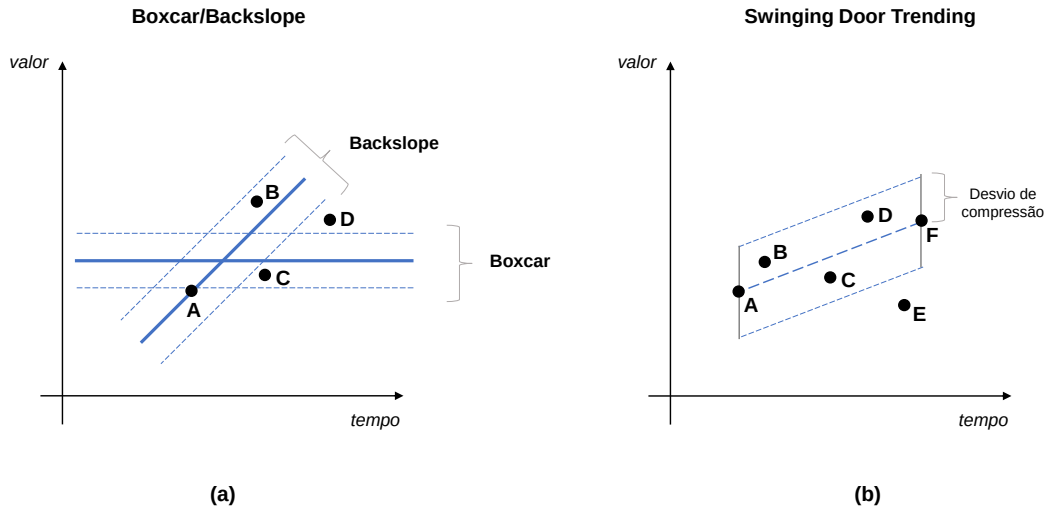

Figura 5.5. Em (a) temos o algoritmo de compressão Boxcar/Backslope, onde o ponto A é o último ponto armazenado, o ponto B viola apenas o Backslope, o ponto C viola apenas o Boxcar e o ponto D viola os dois critérios e portanto é armazenado. Em (b) está o algoritmo Swinging Door Trending, com uma área de cobertura entre o último valor armazenado (A) e o último valor recebido (F), nesse caso os valores B, C e D serão descartados, enquanto o ponto E será armazenado.

5.5. Combinando técnicas de compressão de dados

Diferentes cenários e necessidades de sistemas direcionam para a escolha de um algoritmo de compressão de dados com ou sem perda conforme apresentado acima. A escolha recai sobre a qualidade do dado que se deseja processar, e também no desempenho, custo e eficiência de armazenamento. Geralmente, a compactação com perdas é adequada para dados que não são sensíveis a pequenas alterações ou erros, como arquivos multimídia, conteúdo da Web ou dados analíticos. A compactação com perdas possibilita lidar melhor fatores intrínsecos à transferência de dados, como latência e ocupação de largura de banda, permite economizar mais espaço de armazenamento e acelerar a transferência. No entanto, a compactação com perdas também introduz distorção e degradação dos dados, precisão da análise ou a confiabilidade do backup. A compactação sem perdas é adequada para dados que requerem preservação e reprodução exatas, como arquivos de

texto, arquivos binários ou dados transacionais. A compactação sem perdas pode ajudá-lo a reter todas as informações e a qualidade dos dados e garantir a consistência e a exatidão dos dados. Contudo, a compactação sem perdas também consome mais espaço de armazenamento e recursos e retarda a compactação e descompactação de dados.

Dadas as considerações da Seção 5.4 sobre as técnicas de compressão *lossy* e *lossless* e, retomando os cenários apresentados na Figura 5.1, podemos ponderar sobre quais técnicas são mais aderentes para cada cenário. Para técnica de compressão *lossy*, os cenários 2 e 3 são mais aptos de aplicar, pois alguns tipos de arquivos não necessitam ser restaurados ao tamanho original após terem sido comprimidos. Por exemplo, compressão de imagem padrão JPEG, em que a imagem original tem seu tamanho reduzido e não precisa mais ser restaurado. O mesmo se aplica para arquivos de áudio, no qual o som original pode ser reduzido com baixo impacto ao custo de fidelidade. A compressão do tipo *lossy* elimina apenas sons menos audíveis e menos significativos, reduzindo o espaço necessário para que sejam armazenados ou transmitidos, sem necessidade de restaurar a qualidade original. Contudo para a técnica de compressão *lossless*, apenas o cenário 3 contempla, pois o método de compressão precisa ser revertido através da descompressão, recuperando o arquivo original.

Em alguns casos, pode ser apropriado utilizar os dois tipos de algoritmos de compressão, com e sem perda de dados, como o caso de sistemas de monitoria em *healthcare*, apresentado na Figura 5.6. Nesse contexto, para um diagnóstico confiável é preciso contornar problemas relativos à latência e ocupação da banda, contudo, é um desafio transferir dados coletados periodicamente de diferentes sinais vitais de vários pacientes simultaneamente. Neste cenário, com uma arquitetura em camadas, um algoritmo de compressão *lossy* pode ser aplicado na camada de borda, mais próxima dos sensores, descartando dados redundantes de acordo com cada tipo de sinal vital. Esse estágio de compactação pode ser adaptativo, estabelecendo tempos de aquisição de dados para alguns sinais vitais com pouca variação ao longo do tempo, tais como temperatura e pressão arterial. Por exemplo, a periodicidade da leitura dos sinais pode ser ajustável conforme o estado de saúde da pessoa monitorada. Uma pessoa saudável pode ter uma periodicidade de leitura maior que uma que necessite de acompanhamento com algum tipo de doença. Isso leva a um menor envio de dados mantendo, contudo, o padrão de detecção de falhas. O ajuste adequado desse processo é chave nesse tipo de sistema, um período de leitura muito pequeno leva ao envio de muitos dados, sobrecarregando a rede, gera *overhead*. Contudo, um período muito grande, gera problema de reatividade, no qual se toma uma ação tardia, degradando o resultado esperado.

Passado o estágio adaptativo, os dados dos sinais vitais são então transferidos para o próximo estágio, que aplica o segundo algoritmo, agora *lossless*, comprimindo os dados com métodos tradicionais para reduzir o tamanho total do arquivo antes de enviar para o nuvem, evitando a ocupação desnecessária da largura de banda. Em (Melchiades et al. 2021), os autores apresentam um método de compressão de dados voltado para a I4.0 denominado FastIoT. Nesse método, a partir de um grande volume de dados gerados em dispositivos IoT, a técnica de compressão reduz significativamente o tamanho original possibilitando o envio otimizado das informações, demandando baixa largura de banda de rede, para serem visualizadas de modo ajustado exatamente à região visual do dispositivo de destino. FastIoT funciona dividindo um intervalo de tempo de-

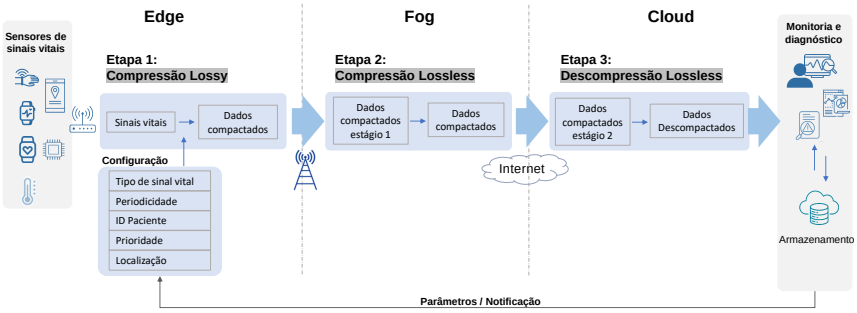

Figura 5.6. Sistema de monitoramento de saúde com duas etapas de compressão. Na camada de borda ocorre a Etapa 1 de compressão, do tipo Lossy. O algoritmo adaptável recebe parâmetros do módulo de configuração e define a cadência da aquisição de dados por tipo de sinal vital e paciente monitorado. Os dados compactados são enviados Etapa 2 de compressão, Lossless. Só então o pacote compactado é enviado pela internet para a nuvem, para ser restaurado, armazenado e utilizado no sistema de monitoração.

finido pelo usuário em períodos iguais, de modo que os valores máximo e mínimo são obtidos e exibidos em um gráfico de barras. Para cada intervalo, o método também retorna sua data inicial, data final e o número de pontos que ele representa. Assim, em termos de comportamento gráfico, o método entrega uma saída composta pelo conjunto de dados original consultado ao servidor. Quanto maior o número de intervalos, melhor a definição do gráfico exibido no lado do cliente.

Desse modo se, por exemplo, dispomos de 1.000 valores IoT armazenados e desejamos plotar em um dispositivo móvel, com uma resolução horizontal de até 500 pontos, não tem necessidade de enviar 1.000 pontos, pois só é possível plotar 500. O método FastIoT se adapta ao destino e entrega de modo eficiente o resultado esperado, ou seja, eficiente na medida em que a janela de transferência de dados se ajusta exatamente à região visual do dispositivo de destino, sem lentidão ou sobrecarga de tráfego de rede. Nestes dois casos analisados, FastIoT e monitoração em sistemas de *healthcare*, os métodos têm como característica principal um processo de compressão adaptável, ou seja, ajustam o nível de compressão através de um mecanismo que considera os principais componentes envolvidos no processo: tipo de dado, sistema de origem, meio de transmissão e sistema de destino. Desse modo, envia apenas o necessário, permitindo um uso eficiente da infraestrutura.

5.6. Métricas para análise de desempenho

A métrica mais básica para avaliar um algoritmo de compressão é a sua taxa de compressão, que consiste na proporção entre o tamanho dos dados compactados e o tamanho dos dados originais, de modo que quanto maior a taxa de compactação, maior o espaço poupado. Entretanto, uma alta taxa de compressão pode resultar alta perda de informações, portanto, a taxa de compactação deve ser considerada juntamente com outras métricas que medem a qualidade e a precisão dos dados compactados. Além disso, outros elementos de análise podem ser uma ferramenta bastante útil para avaliar eficiência e também possibilitar uma melhor comparação entre diferentes métodos. Para medir a

Taxa de Compressão (TC), temos a divisão do tamanho não compactado pelo tamanho compactado (veja Equação 3).

$$TC = \frac{amanho\ dados\ descompactados}{Tamanho\ dados\ compactados} \tag{3}$$

A métrica de Taxa de Erro de Bits, do inglês *Bit Error Rate* (BER), é a medida da integridade dos dados após o processo de descompactação. Este índice baixo indica uma alta fidelidade dos dados compactados, com o algoritmo de compactação preservando a integridade dos bits originais. Contudo, BER alto indica uma baixa precisão dos dados compactados, com o algoritmo de compactação gerando um resultado que contém distorções. Esse índice faz sentido quando aplicado aos métodos de compressão sem perdas, *lossless*, na medida em que os métodos de compressão com perdas, naturalmente os dados descompactados divergem dos originais. O cálculo de BER é dado pelo número de bits do arquivo descompactado com erro dividido pelo total de bits do arquivo original.

Outra métrica útil para avaliação de compressão com perdas é a distorção, que quantifica a qualidade do sinal reconstruído. A medida de distorção mais utilizada é a Diferença da Raiz Quadrática Média, do inglês *Root-Meal-Square Deviation* (RMSD), dado percentualmente pela Equação 4. Aqui, temos a distorção $d(x,\hat{x})$, entre os vetores x e $\hat{x} \in \mathbf{R}$, sendo x o sinal original e $\hat{x}$ o sinal restaurado após a descompressão. Além dessas medidas para avaliar o desempenho do algoritmo de compressão, a solução fi-a-fim precisa ser analisada considerando também fatores que podem afetar o resultado final do processo de compressão. Conforme explorado na Seção 5.3 os fatores tempo de compressão, descompressão e armazenamento influenciam diretamente no desempenho e qualidade final em um processo de compressão.

$$PRMSD = \sqrt{\sum_{n=1}^{N} \frac{(x[n] - \hat{x}[n])^2}{x[n]^2}} \times 100 \tag{4}$$

5.7. Conclusão

Esse capítulo de livro trouxe a tona o tema de compressão de dados, que pode ser explorado em sistemas computacionais que demandam alto desempenho. Mais precisamente, hoje dados são vistos como a grande moeda do século XXI. Tecnologias como sensores, atuadores, GPU (Graphical Processing Unit), bem como algoritmos de inteligência artificial fazem com que tenhamos uma curva exponencial positiva na geração de dados. Para tratá-los de forma efetiva, são necessárias estratégias que garantam a escalabilidade de sistemas de computação, oferecendo qualidades de serviço aceitáveis para usuários finais independente do número deles e da carga de trabalho em voga.

A compressão, portanto, atua como um meio de diminuir o tamanho dos arquivos que serão trafegados pela rede, podendo também fazer com que dados de tamanho menor sejam armazenados. Isso propicia uma melhor utilização da largura de banda da rede, principalmente no tocante a redes com alta flutuação como pode ser a Internet, bem como o armazenamento de uma quantidade maior de dados em memória secundária. Diante disso, esse capítulo de livro apresentou um formalismo matemático para modelar o uso de

compressão de dados, bem como diferentes técnicas e seu casamento para diferentes tipos de arquivos e demandas. Como mencionado ao longo do documento, compressão não é um assunto novo, mas ganha momento frente as direções de foco em que vivemos. Além disso, questões arquiteturais de memória cache, pipeline, TLP (Thread Level Parallelism), coprocessadores matemáticos parecem estar bem resolvidas e sedimentadas e as técnicas aqui apresentadas se mostram como uma alternativa para gerir sistemas escaláveis e com alta disponibilidade.

Agradecimentos

Os autores gostariam de agradecer aos seguintes órgãos de fomento: Fundação de Amparo a Pesquisa do Estado do Rio Grande do Sul (FAPERGS, processo 21/2551-0000118-6); Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, processo 305263/2021-8).

Referências

Abuin et al. 2022 Abuin, A., Iradier, E., Fanari, L., Montalban, J., and Angueira, P. (2022). High efficiency wireless-noma solutions for industry 4.0. In *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*, pages 1–8. IEEE.

Aggarwal and Lan 2020 Aggarwal, V. and Lan, T. (2020). Modeling and optimization of latency in erasure-coded storage systems. *arXiv preprint arXiv:2005.10855*.

Aheleroff et al. 2020 Aheleroff, S., Xu, X., Lu, Y., Aristizabal, M., Velásquez, J. P., Joa, B., and Valencia, Y. (2020). Iot-enabled smart appliances under industry 4.0: A case study. *Advanced engineering informatics*, 43:101043.

Berardinelli et al. 2018 Berardinelli, G., Mahmood, N. H., Rodriguez, I., and Mogensen, P. (2018). Beyond 5g wireless irt for industry 4.0: Design principles and spectrum aspects. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE.

Castiglione et al. 2021 Castiglione, A., Umer, M., Sadiq, S., Obaidat, M. S., and Vijayakumar, P. (2021). The role of internet of things to control the outbreak of covid-19 pandemic. *IEEE Internet of Things Journal*, 8(21):16072–16082.

Chen et al. 2020 Chen, W.-Y., Chou, P.-Y., Wang, C.-Y., Hwang, R.-H., and Chen, W.-T. (2020). Live video streaming with joint user association and caching placement in mobile edge computing. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 796–801. IEEE.

Das and Rahman 2021 Das, S. K. and Rahman, M. Z. (2021). A compression technique for electronic health data through encoding. In *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–6. IEEE.

Dwivedi et al. 2022 Dwivedi, S. K., Yadav, J., Ansar, S. A., Khan, M. W., Pandey, D., and Khan, R. A. (2022). A novel paradigm: Cloud-fog integrated iot approach. In *2022 3rd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, pages 1–5. IEEE.

Elkaseer et al. 2018 Elkaseer, A., Salama, M., Ali, H., and Scholz, S. (2018). Approaches to a practical implementation of industry 4.0. *Resource*, 3:5.

Fischer et al. 2020 Fischer, G. S., da Rosa Righi, R., de Oliveira Ramos, G., da Costa, C. A., and Rodrigues, J. J. (2020). Elhealth: Using internet of things and data prediction for elastic management of human resources in smart hospitals. *Engineering Applications of Artificial Intelligence*, 87:103285.

Gia et al. 2019 Gia, T. N., Qingqing, L., Queralta, J. P., Tenhunen, H., Zou, Z., and Westerlund, T. (2019). Lossless compression techniques in edge computing for mission-critical applications in the iot. In *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pages 1–2. IEEE.

Idrees and Khlief 2023 Idrees, A. K. and Khlief, M. S. (2023). Efficient compression technique for reducing transmitted eeg data without loss in iomt networks based on fog computing. *The Journal of Supercomputing*, pages 1–26.

Khan et al. 2020 Khan, M. A., Pierre, J. W., Wold, J. I., Trudnowski, D. J., and Donnelly, M. K. (2020). Impacts of swinging door lossy compression of synchrophasor data. *International Journal of Electrical Power & Energy Systems*, 123:106182.

Khlief and Idrees 2022 Khlief, M. S. and Idrees, A. K. (2022). Efficient eeg data compression technique for internet of health things networks. In *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pages 403–409. IEEE.

Lu et al. 2020 Lu, T., Xia, W., Zou, X., and Xia, Q. (2020). Adaptively compressing iot data on the resource-constrained edge. In *HotEdge*.

Marinagi et al. 2023 Marinagi, C., Reklitis, P., Trivellas, P., and Sakas, D. (2023). The impact of industry 4.0 technologies on key performance indicators for a resilient supply chain 4.0. *Sustainability*, 15(6):5185.

Melchiades et al. 2021 Melchiades, M. B., Crovato, C. D. P., Nedel, E., Schreiber, L. V., and Righi, R. D. R. (2021). Fastiot: an efficient and very fast compression model for displaying a huge volume of iot data in web environments. *International Journal of Grid and Utility Computing*, 12(5-6):605–617.

Panwar 2020 Panwar, S. (2020). Breaking the millisecond barrier: Robots and self-driving cars will need completely reengineered networks. *IEEE Spectrum*, 57(11):44–49.

Park et al. 2020 Park, J., Samarakoon, S., Shiri, H., Abdel-Aziz, M. K., Nishio, T., Elgabli, A., and Bennis, M. (2020). Extreme urllc: Vision, challenges, and key enablers. *arXiv preprint arXiv:2001.09683*.

Roy and Nikolaidis 2022 Roy, S. K. and Nikolaidis, I. (2022). Limited size lossy compression for wsns. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pages 359–362. IEEE.