Um estudo experimental sobre estratégias de coleta de dados em pipelines meteorológicos

Hiara F. Gonçalves[1], Gabriel C. R. T. Silva[1], Vinícius Viana[1], Flávio B. da Silva Mota[2], Vanessa C. O. Souza[1], Melise Maria V. de Paula[1]

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Av. BPS, 1303 – Prédio C – Pinheirinho, Itajubá – MG – Brazil

[2]Descubra Soluções em Decisões Estratégicas
Av. BPS, 1303 – Prédio J3 – Pinheirinho, Itajubá – MG – Brazil

{d2023005010,d2023005594,vanessasouza,melise}@unifei.edu.br,

{viniciusvsv193,flavio.belizario.mota}@gmail.com

Abstract. *Meteorological data is essential for various applications, making efficient data collection a critical factor in data pipelines. This work presents a comparative study of three strategies for accessing ECMWF data: the official Web API, direct HTTP collection, and S3-compatible object storage access. The goal is to identify the most efficient approach in scenarios involving frequent updates. Controlled experiments were conducted using Apache JMeter to evaluate performance metrics such as response time, throughput, and resource consumption under concurrent load. The results aim to guide the selection of data collection architectures that optimize the performance and reliability of operational meteorological pipelines.*

Resumo. *Dados meteorológicos são essenciais para diferentes tipos de aplicações, tornando a eficiência na coleta um fator crítico em pipelines de dados. Este trabalho apresenta um estudo comparativo entre três estratégias de acesso aos dados do ECMWF: a Web API oficial, a coleta via HTTP e o acesso por objetos compatíveis com S3. O objetivo é identificar a abordagem mais eficiente em cenários de atualizações frequentes. Foram conduzidos experimentos controlados com o Apache JMeter, para avaliar métricas de tempo, throughput e consumo de recursos sob carga concorrente. Os resultados visam orientar a escolha de arquiteturas de coleta que otimizem o desempenho e a confiabilidade de pipelines meteorológicos operacionais.*

1. Introdução

Dados meteorológicos são fundamentais para aplicações em previsão do tempo, monitoramento ambiental e suporte à tomada de decisão em diversos setores. Entre os principais provedores globais desses dados está o ECMWF (*European Centre for Medium-Range Weather Forecasts*), uma organização internacional responsável pela produção de previsões meteorológicas numéricas globais e pela manutenção de um dos maiores repositórios de dados meteorológicos do mundo [ECMWF 2026a]. Além da relevância científica desses produtos e serviços, a obtenção eficiente desses dados é essencial para diferentes aplicações reais que dependem de atualizações frequentes.

O ECMWF disponibiliza um subconjunto de seus dados por meio do pacote *ECMWF Open Data* em formato GRIB2, com resolução espacial de aproximadamente 0,25 graus [ECMWF 2026c]. Embora eficiente para a compactação e a representação de dados multidimensionais, esse formato demanda ferramentas específicas de manipulação e pode impor custos relevantes de transferência e de processamento em pipelines operacionais [ECMWF 2024b].

Devido ao grande volume desses arquivos e à natureza operacional das previsões meteorológicas, a eficiência na coleta desses dados é um fator crítico. Atrasos nessa etapa podem comprometer a atualidade das informações e impactar sistemas dependentes de previsões atualizadas [Bauer et al. 2015]. No contexto deste trabalho, coleta de dados refere-se à etapa de obtenção dos arquivos meteorológicos diretamente da fonte, por meio de APIs, requisições HTTP ou acesso a serviços de armazenamento de objetos (*object storage*), nos quais os dados são organizados como objetos identificados por chaves em estruturas do tipo *Bucket* (contêiner lógico utilizado para agrupar objetos). Essa etapa precede a ingestão completa no pipeline e influencia diretamente o tempo de disponibilização dos dados para processamento [Kleppmann 2017].

Diferentes estratégias podem ser empregadas para obter dados meteorológicos do ECMWF em ambientes computacionais, incluindo acesso programático via API Web, download direto via HTTP e recuperação a partir de interface compatível com a API do S3 [ECMWF 2026c]. Neste trabalho, essas estratégias são avaliadas como alternativas para a etapa de coleta de dados em pipelines meteorológicos.

Apesar da existência de múltiplas estratégias de acesso, ainda são escassos estudos comparativos que avaliem, sob um mesmo ambiente experimental, o desempenho dessas alternativas para coleta de dados meteorológicos. Em particular, há pouca evidência consolidada sobre como diferentes mecanismos de acesso se comportam em termos de tempo total de transferência, *throughput* e consumo de recursos computacionais sob diferentes condições de carga. Nesse contexto, estudos de *benchmarking* tornam-se relevantes para identificar gargalos e apoiar decisões de projeto em pipelines de dados [Pargaonkar 2023].

Diante disso, este trabalho apresenta uma análise comparativa de desempenho entre três estratégias de coleta de dados meteorológicos: acesso por meio de Web APIs, download direto via HTTP e acesso a serviços de armazenamento de objetos. O objetivo é identificar qual abordagem apresenta melhor desempenho em cenários de coleta de dados do ECMWF, contribuindo tanto para o projeto de pipelines mais eficientes quanto para a produção de evidências experimentais sobre esse problema.

Os resultados indicam, em linhas gerais, diferenças relevantes entre as abordagens, com implicações práticas para o projeto de pipelines meteorológicos orientados a desempenho. O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico, a Seção 3 descreve a metodologia experimental, a Seção 4 discute os resultados e a Seção 5 apresenta as conclusões.

2. Referencial

Esta seção apresenta os conceitos fundamentais para a compreensão do problema investigado, incluindo o formato dos dados, os mecanismos de acesso e os critérios de avaliação de desempenho. O formato GRIB2 (*General Regularly-distributed Information in Binary*

Edition 2) é um padrão da Organização Meteorológica Mundial (WMO) para armazenamento e transporte de dados meteorológicos em grade. Um arquivo GRIB2 é uma sequência contínua de blocos binários chamados mensagens, cada mensagem é independente e autodescritiva, estruturada em *Sections* que vão de 1 até 8 [Heise 2019]. Devido a essa organização, seu uso é muito difundido em sistemas de previsão numérica, já que um arquivo pode conter milhares de campos empilhados, o que combina uma representação compacta de campos multidimensionais com suporte à disseminação operacional de grandes volumes de dados [WMO 2019].

Em contrapartida, essa eficiência de compactação não elimina os desafios de acesso e de processamento: a organização interna dos arquivos em mensagens sequenciais dificulta a recuperação parcial de conteúdo e pode impor custos significativos de transferência e de pós-processamento em pipelines operacionais [Gowan et al. 2022b]. Esse problema fica ainda mais relevante com o aumento da resolução espacial e temporal dos modelos, que produzem volumes crescentes de dados e demandam mecanismos de coleta compatíveis com cenários de baixa latência [Schwitalla et al. 2017].

O ECMWF disponibiliza previsões geradas pelos modelos IFS (*Integrated Forecasting System*), baseados em modelagem física da atmosfera, e AIFS (*Artificial Intelligence Forecasting System*), baseados em aprendizado de máquina. Esses produtos são distribuídos por meio de arquivos GRIB2. Os dados operacionais (operational IFS) são produzidos continuamente em ciclos fixos (00, 06, 12, 18 UTC), usados para previsão em tempo quase real, apresentam alta confiabilidade, estabilidade e SLA definidos e são distribuídos com baixa latência. Ademais, o ecossistema do ECMWF disponibiliza dados não operacionais, como as reanálises ERA5, onde o objetivo é focar em qualidade científica, não em latência e, por isso, os dados não são "previsões em tempo real".

Diferentes modelos de acesso têm sido adotados pelos provedores de dados meteorológicos. O acesso via API Web oferece uma camada de abstração que permite ao cliente especificar subconjuntos de interesse, como variáveis e níveis atmosféricos, atribuindo ao servidor a seleção e a preparação dos dados solicitados. No caso do ECMWF, essa abordagem pode ser realizada por meio do sistema MARS (*Meteorological Archival and Retrieval System*), reduzindo o volume transferido, mas gerando custos adicionais associados ao processamento remoto e ao tempo de fila [ECMWF 2024a].

No contexto deste trabalho, o acesso via API ocorre por meio de *datasets* específicos disponibilizados pelo ECMWF, como S2S e TIGGE. O S2S (*Sub-seasonal to Seasonal*) é voltado a previsões sub-sazonais e sazonais, enquanto o TIGGE (*THORPEX Interactive Grand Global Ensemble*) reúne previsões por conjunto de diferentes centros meteorológicos, com atraso de 48 horas no acesso público [ECMWF 2026d, ECMWF 2026e].

A transferência direta via HTTP representa uma abordagem mais próxima da recuperação do arquivo bruto, normalmente com menor complexidade de interação e menor sobrecarga de processamento no lado do provedor. Em contrapartida, essa estratégia pode implicar desperdício de banda, já que o cliente frequentemente precisa transferir mensagens GRIB2 que não serão utilizadas.

Já o acesso por armazenamento de objetos compatível com S3 constitui uma alternativa orientada à escalabilidade, sendo frequentemente adotado em cenários de aquisição de grandes volumes de dados e integração com pipelines distribuídos

[Khemka and Raj 2025]. No ecossistema do ECMWF, os mecanismos via HTTP e S3 viabilizam o acesso a dados operacionais recentes, isto é, previsões produzidas em ciclos regulares e disponibilizadas com baixa latência, caracterizando acesso em regime de tempo quase real (*near real-time*) [ECMWF 2026b].

Uma alternativa para mitigar o custo de transferência de arquivos GRIB2 volumosos é a coleta seletiva de dados. Essa estratégia baseia-se na capacidade do servidor de responder a requisições parciais de um arquivo binário, sem necessidade de transferi-lo integralmente. Tal mecanismo é suportado por meio de *HTTP Range Requests*, definidos pela RFC 9110, que permitem ao cliente requisitar intervalos específicos de bytes. Nesses casos, o servidor responde com o código HTTP `206 (Partial Content)` [Fielding et al. 2022].

A aplicação dessa técnica depende da existência de um arquivo de índice (`.idx`), que funciona como um mapa de metadados contendo a posição, em bytes, de cada mensagem GRIB2 no arquivo principal. A partir do *parsing* desse índice, torna-se possível automatizar a recuperação seletiva de variáveis, eliminando o desperdício de banda e o esforço computacional de pós-processamento local [Gowan et al. 2022a].

A eficiência dessa abordagem, no entanto, não depende apenas da redução do volume de dados. Trabalhos sobre avaliação de desempenho em sistemas distribuídos mostram que o comportamento observado durante a transferência resulta da interação entre latência de rede, processamento de metadados, concorrência e consumo de recursos locais [Rao et al. 2019, Bocchi et al. 2014]. Assim, a análise de desempenho de pipelines de coleta não deve se restringir ao tempo total de transferência, mas considerar também métricas como *throughput*, uso de CPU e consumo de memória, sobretudo em cenários de carga concorrente e alta frequência de acesso [Saeedizade et al. 2023].

Desta forma, o *benchmarking* representa uma estratégia adequada para a comparação desses métodos de acesso, pois permite avaliar aspectos como a escalabilidade, a identificação de gargalos e o comportamento sob diferentes condições operacionais [Pargaonkar 2023]. Para este trabalho, essas métricas são particularmente relevantes por permitirem avaliar não apenas a velocidade de transferência, mas também o custo computacional associado a cada estratégia de coleta, o que é essencial em pipelines meteorológicos que exigem atualizações frequentes e disponibilidade contínua dos dados.

Em um estudo técnico comparativo realizado, analisou-se a recuperação de dados científicos que evidenciou a superioridade de objetos em nuvem em termos de *throughput* e escalabilidade horizontal para dados científicos [Gallagher et al. 2024]. O estudo aponta que, enquanto as APIs tradicionais tendem a saturar em aproximadamente 100MB/s devido a sobrecarga de protocolo e processamento no servidor, o acesso direto a objetos em formatos otimizados pode atingir taxas 100 vezes superiores. No entanto, protocolos baseados em HTTP com suporte a *Range Requests* surgem como uma alternativa intermediária eficiente, equilibrando a facilidade de implementação com a capacidade de recuperação parcial dos arquivos. Essas evidências teóricas ressaltam a importância de validar tais métricas em contextos operacionais específicos, como dos dados do ECMWF.

É válido ressaltar que este estudo tem o objetivo de investigar, no contexto específico do ECMWF, qual estratégia de coleta apresenta melhor desempenho em nosso pipeline operacional, sendo essa a fonte de dados de interesse. Além disso, o ECMWF

oferece múltiplos mecanismos de acesso a dados meteorológicos, como apresentados anteriormente, o que permite uma comparação experimental controlada entre diferentes estratégias. Ferramentas de IA generativa foram aplicadas para a revisão linguística e auxílio na estruturação deste estudo. O experimento, coleta de dados e análise dos resultados foram conduzidos integralmente pelos autores, que assumem total responsabilidade pelo conteúdo apresentado.

3. Desenvolvimento

O objetivo principal dos testes realizados foi avaliar os três métodos distintos de transferência de dados *open* do ECMWF. Os testes foram executados a partir de um cliente localizado no Brasil, conectando-se a servidores do ECMWF na Europa. Na metodologia utilizada para o desenvolvimento do trabalho, foram definidas 5 etapas, conforme ilustra a Figura 1.

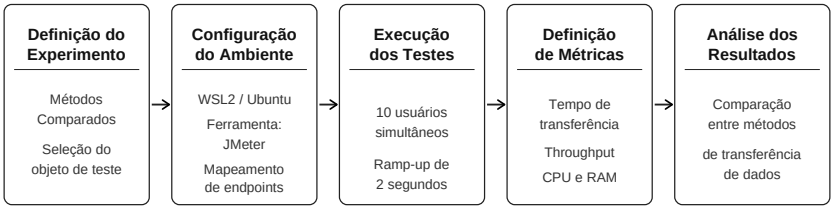

Figura 1. Etapas da Metodologia

Na primeira etapa, Definição do Experimento, foram estabelecidos os elementos iniciais do estudo, incluindo os métodos comparados e a seleção dos dados para os testes. Essa fase abrangeu o planejamento dos testes e a definição do que seria avaliado. Com a seleção dos métodos a serem comparados, foram definidos os dados considerando o modelo IFS.

A segunda etapa, Configuração do Ambiente, descreve a infraestrutura utilizada na execução dos testes. Os experimentos foram conduzidos em ambiente Linux virtualizado com WSL2, com Ubuntu 22.04.5 LTS. A máquina utilizada possui processador Intel Core i5-10210U de 1,6 GHz, com 1 núcleo físico, 2 *threads* alocadas para a máquina virtual e 3,8 GB de memória RAM.

O sistema de armazenamento baseia-se em disco rígido rotacional (HDD), fator considerado na análise por seu possível impacto na latência de escrita e, consequentemente, no tempo total de transferência dos dados. Para a automação dos testes e geração de carga, foi utilizada a ferramenta Apache JMeter [Apache Software Foundation 2026]. Como não foram configuradas políticas de *retry*, qualquer requisição que apresentou erro foi imediatamente contabilizada como falha. Dessa forma, a resiliência reportada reflete o comportamento intrínseco dos métodos avaliados, sem a interferência de mecanismos externos de recuperação da aplicação. Além disso, foram mapeados os endpoints de comunicação de cada método e foi obtida a chave de conexão com a API.

Na terceira etapa Execução dos testes, definiu-se a carga de trabalho com grupos de 10 usuários simultâneos e um período de *ramp-up* de 2 segundos. O experimento foi dividido em dois cenários: transferência integral dos dados e a coleta seletiva. No primeiro cenário, Cenário 1, o objetivo foi avaliar o desempenho bruto de transferência dos

arquivos meteorológicos GRIB2 em sua forma integral. Foram analisados dois métodos de acesso: direto via HTTP utilizando o *endpoint* `data.ecmwf.int` com uma requisição do tipo `GET`. O outro método avaliado foi o acesso via *Object Storage* com o *endpoint* do *Bucket* na região `eu-central-1` da AWS, também via requisição `GET`. A figura 2 ilustra o fluxo de execução para a transferência dos dados neste cenário:

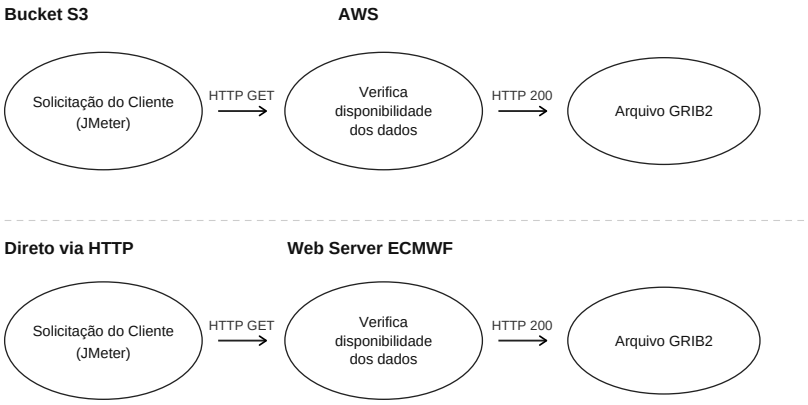

Figura 2. Fluxo de Transferência de Dados

Para assegurar a comparabilidade entre os métodos de acesso avaliados, os testes foram configurados para transferir o mesmo arquivo. Para isso, foram fixados a mesma data de referência, o mesmo `step` de 0 horas, que corresponde ao horizonte de previsão coincidente com o instante inicial do modelo, e o mesmo produto do modelo IFS operacional com resolução espacial de 0,25°. Adicionalmente, os experimentos foram realizados em três períodos distintos do dia (manhã, tarde e noite), ao longo de quatro dias, com o objetivo de observar possíveis variações de desempenho associadas a flutuações temporais da rede e do *throughput*.

Em ambos os métodos, o JMeter envia uma requisição `HTTP GET` para a URL específica do arquivo a ser transferido. No acesso direto via HTTP, essa requisição é encaminhada ao servidor web do ECMWF. Já no acesso via *Bucket*, a requisição é direcionada à infraestrutura de armazenamento da AWS. Após a verificação da disponibilidade do arquivo, o fluxo é concluído com o retorno da resposta `HTTP 200 (OK)`, indicando o início da transferência do conteúdo solicitado. A Web API não foi incluída nesse cenário já que ela foi projetada para operações de recorte e filtragem o que exige processamento intensivo do servidor antes de disponibilizar os dados, o que causaria um *overhead* de processamento e tempo de fila excessivo.

O segundo cenário foi elaborado para representar uma aplicação operacional em que não há necessidade de transferir ou processar o arquivo completo. Desta forma, toda a lógica de seleção de dados é deslocada para a camada de rede ou para o servidor, reduzindo a necessidade de pós-processamento local e, potencialmente, o consumo de banda e de armazenamento. Para essa análise, foram selecionadas três variáveis meteorológicas: temperatura a 2 metros, precipitação total e pressão à superfície. A fim de garantir a comparabilidade entre os métodos avaliados, foram mantidos os mesmos valores de data e de *step* em todos os casos.

O objetivo desse cenário foi analisar o impacto da requisição seletiva no tempo de

resposta e no volume de dados transferidos. A escolha por variáveis de nível de superfície se justifica pela necessidade de garantir a paridade experimental entre os métodos, já que a Web API apresenta restrições na disponibilidade de determinados níveis de pressão em comparação aos arquivos operacionais completos. Assim como no primeiro cenário, os testes foram executados em diferentes períodos do dia, ao longo de quatro dias, com o propósito de identificar possíveis variações temporais no desempenho. A Figura 3 apresenta o fluxo correspondente a esse experimento no caso da API.

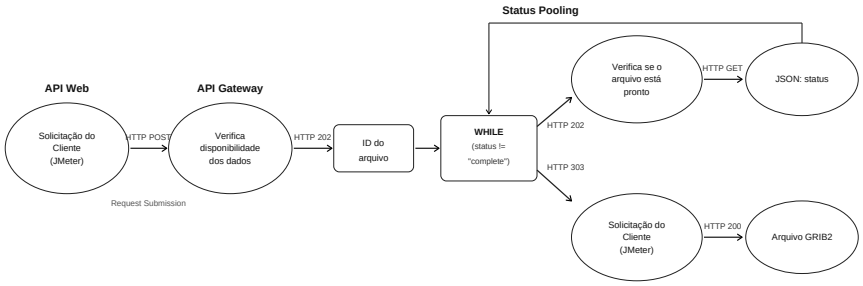

Figura 3. Etapas do Teste Seletivo da API Web Oficial (JMeter)

A requisição inicial foi realizada enviando um corpo de requisição em formato JSON ao *endpoint* `api.ecmwf.int` correspondente ao repositório de dados TIGGE, com o método `HTTP POST`. O campo `param` foi utilizado para especificar as três variáveis de interesse, por meio dos códigos 167, 228 e 134, correspondentes às variáveis no nível de superfície (`sfc`). Com essa configuração, o sistema MARS realiza a seleção e a concatenação das mensagens GRIB antes de disponibilizar o arquivo resultante. Para assegurar a comparabilidade com os testes realizados via *bucket* do S3 e acesso direto via HTTP, a requisição foi configurada com a classe operacional (`class:od`) e o fluxo operacional (`stream:oper`) . Abaixo está o JSON feito para a requisição:

```
{
  "class": "od",
  "date": "2026-03-21",
  "expver": "1",
  "levtype": "sfc",
  "param": "167/134/228",
  "step": "0",
  "stream": "oper",
  "type": "fc"
}
```

A escolha do *dataset* TIGGE se deu por ser o único ponto de acesso via Web API do ECMWF que permite coletar dados operacionais do modelo IFS de forma programática e autenticada, viabilizando a comparação entre os métodos. Por fim, o campo `step`, definido como 0, e o parâmetro `type:fc`, que indica um dado de previsão (*forecast*), complementam a especificação da solicitação. Após o envio da requisição, o servidor retorna o código `202 (Accepted)`, indicando que o pedido foi aceito, e encaminha para a fila de processamento do ECMWF. Nessa etapa, também é retornado um identificador único da solicitação, extraído dinamicamente por um `Post-Processor`. Esse identificador é utilizado como referência nas consultas subsequentes ao estado da requisição.

A partir disso, inicia-se uma rotina de *polling*, na qual o JMeter realiza chamadas periódicas, enviando esse `ID`, até que o status da solicitação indique a disponibilidade do dado. Quando o status retorna `'completed'`, o ciclo de consultas é encerrado e a carga do arquivo é iniciada, concluindo a transferência via API Web. Na figura 4, está representado o fluxo para os testes utilizando o *Bucket* e a transferência via HTTP:

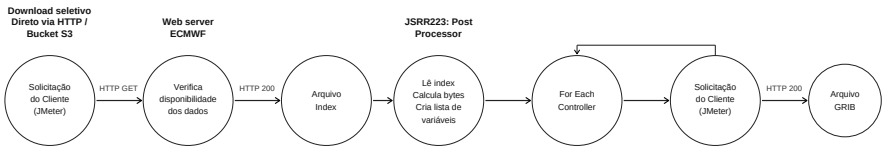

Figura 4. Etapas do Teste Seletivo Bucket S3 e direto via HTTP (JMeter)

A estrutura de diretórios do *Bucket* foi analisada para avaliar a viabilidade de filtragem nativa no processo de recuperação de dados. Constatou-se que, embora o Amazon S3 disponibilize mecanismos de seleção, como o `S3 Select`, esse recurso é restrito a formatos estruturados simples, como CSV e JSON, não sendo compatível com a organização binária dos arquivos GRIB2. Assim, não foi possível utilizar essa funcionalidade para recuperar diretamente apenas as variáveis de interesse. Diante dessa limitação, a estratégia adotada para avaliar o acesso seletivo via *Bucket* consistiu em uma abordagem híbrida baseada no uso do arquivo de índice.

O arquivo de índice disponibilizado pelo ECMWF segue o formato JSON Lines (JSONL), no qual cada linha representa uma mensagem GRIB2 do arquivo principal e contém metadados como data, tipo de previsão, variável e dois campos essenciais para a recuperação seletiva: `_offset`, que indica a posição em bytes onde a mensagem começa no arquivo GRIB2, e `_length`, que indica o tamanho em bytes dessa mensagem. O `JSR223 PostProcessor` realiza o *parsing* linha a linha desse arquivo, filtrando as entradas cujo campo `param` corresponde às variáveis de interesse. Para cada variável identificada, o intervalo de bytes é calculado a partir dos campos `_offset` e `_length`, sendo o byte final dado por `_offset + _length - 1`. Esse intervalo é adicionado a uma lista percorrida pelo `ForEach Controller`, que emite uma requisição HTTP individual para cada variável com o cabeçalho `Range: bytes=início-fim`, ao qual o servidor responde com `HTTP 206 (Partial Content)`.

Na quarta etapa, Definição de Métricas, foram definidos os indicadores de desempenho a serem analisados, incluindo o tempo de transferência, *throughput*, uso de CPU e de memória RAM. O monitoramento contínuo e sincronizado do uso percentual de CPU e do consumo de memória RAM foi realizado com o *plugin* PerfMon Metrics Collector, com intervalo de coleta de 1 segundo. As métricas de rede, como *throughput* e tempo de resposta (*elapsed time*), foram extraídas diretamente dos *logs* de amostragem gerados pelo JMeter, permitindo a obtenção das medidas de forma consistentes para a análise de desempenho. Na última etapa, Análise dos resultados, as métricas coletadas foram analisadas considerando os dois cenários definidos.

4. Resultados

Nesta seção, serão apresentados os resultados obtidos com os testes realizados no JMeter. A análise será dividida em duas etapas: cenário 1 e cenário 2, correlacionando as métricas de rede com o recurso computacional local. Os resultados completos podem ser encontrados na planilha disponibilizada[1]. Conforme já descrito, o cenário 1 avaliou a capacidade dos protocolos de lidar com a transferência completa dos arquivos, com um tamanho médio de 120MB cada. Na tabela 1 estão as médias dos resultados obtidos nos testes:

Tabela 1. Métricas coletadas com o Cenário 1

Método	Período	Média do Tempo (s)	Desvio Padrão	Throughput
Bucket S3	Manhã	89,57	12,79	5,50 req/min
URL direta	Manhã	75,58	4,52	7,17 req/min
Bucket S3	Tarde	131,82	19,43	4,60 req/min
URL direta	Tarde	130,00	9,06	5,70 req/min
Bucket S3	Noite	151,17	22,88	4,22 req/min
URL direta	Noite	153,93	8,75	5,07 req/min

Os resultados indicam que o período da manhã apresentou maior eficiência para ambos os métodos. O acesso via URL direta registrou a menor latência (75,58 s), sendo superior ao *Bucket* S3 (89,57 s) nesse intervalo. Outra característica é que em todos os períodos o método direto via HTTP manteve um desvio padrão muito menor em comparação ao S3, que chegou a registrar 22,88 s no período noturno, sendo o maior de todos. Para essa etapa do pipeline dos dados meteorológicos, a menor variabilidade da URL direta é preferível, pois permite um planejamento mais previsível do tempo.

Observou-se uma degradação progressiva no desempenho dos métodos à medida que o dia avançava. Esse comportamento sugere que o gargalo não reside apenas no provedor (ECMWF ou AWS), mas possivelmente no tráfego de rede global e na latência de escrita no hardware local (HDD), que atua como limitador físico para transferências volumosas. A queda no *throughput* confirma essa perda de eficiência.

Durante a execução dos testes, registrou-se erros *Too Many Requests* (HTTP 429) principalmente no período da noite. Essas falhas podem explicar o comportamento dos resultados obtidos nesse período e indicam que a carga de 10 usuários simultâneos baixando os arquivos atingiu o limite da taxa imposta pela infraestrutura do provedor. É importante destacar que as médias apresentadas na tabela correspondem apenas às requisições finalizadas com sucesso. Entretanto, a presença desses erros indica que a transferência integral é um método de baixa resiliência em cenários de alta concorrência. As Figuras 5 e 6 apresentam os gráficos de consumo de recursos do método direto via HTTP em dois períodos distintos.

O monitoramento dos recursos computacionais revelou perfis de consumo distintos, reforçando os dados obtidos. No período da manhã, o gráfico indica um comportamento mais estável. A CPU se mantém abaixo de 20% e a transferência é concluída em 82 segundos. Em contrapartida, o gráfico do período da noite evidencia o gargalo observado.

[1] `https://docs.google.com/spreadsheets/d/1K89mRThiHR9hNwKwDR8_DMmMjDnJUqbUFWFMNssb1P0/edit?usp=sharing`

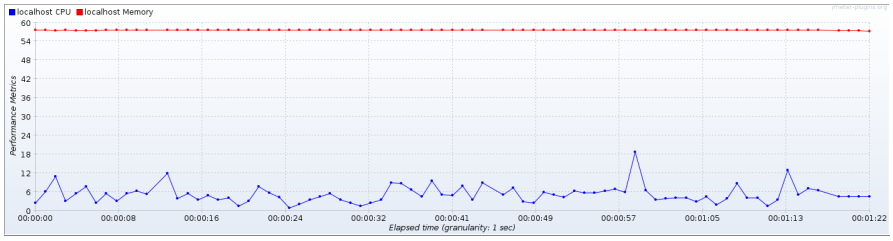

Figura 5. Consumo de CPU e Memória - método direto via HTTP (Manhã). A linha azul indica o consumo de CPU e a vermelha o consumo de memória.

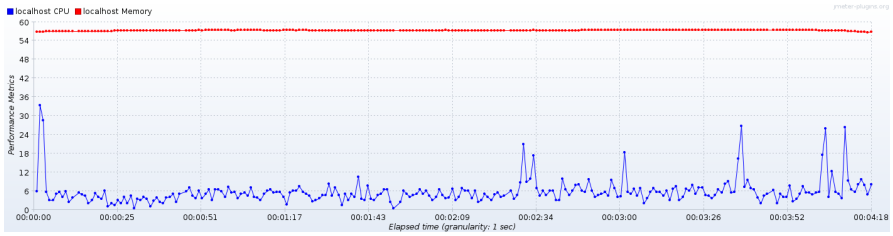

Figura 6. Consumo de CPU e Memória - método direto via HTTP (Noite). A linha azul indica o consumo de CPU e a vermelha o consumo de memória.

É possível ver oscilações no consumo de CPU e a conclusão da transferência em pouco mais de 4 minutos. Isso sugere um esforço maior do sistema e tempos de espera mais longos. O cenário 2, de acordo com o que foi descrito, analisou a transferência seletiva dos dados e observou-se uma diminuição drástica no tamanho dos arquivos, que antes eram 120MB, com as três variáveis selecionadas foi em média 2MB. Abaixo na tabela 2 estão as médias dos dados coletados no JMeter:

Tabela 2. Métricas coletadas com o Cenário 2

Método	Período	Média do Tempo (s)	Desvio Padrão	Throughput
API Web	Manhã	9,90	0,93	0,49 req/seg
Bucket S3	Manhã	2,21	0,61	5,10 req/seg
URL	Manhã	1,90	0,43	5,37 req/seg
API Web	Tarde	11,25	2,08	0,31 req/seg
Bucket S3	Tarde	1,98	0,44	5,82 req/seg
URL	Tarde	1,77	0,46	5,9 req/seg
API Web	Noite	10,69	1,46	0,41 req/seg
Bucket S3	Noite	1,84	0,41	5,45 req/seg
URL	Noite	1,65	0,45	5,55 req/seg

Os resultados obtidos revelam um contraste nítido entre a Web API oficial e os métodos baseados no arquivo de índice. Enquanto a API apresentou tempos médios superiores a 10 segundos em todos os períodos, o acesso via S3 e URL direta se mantiveram com médias entre 1,65 e 2,21 segundos, representando uma aceleração na entrega dos dados. O método via HTTP direto se destacou como o mais eficiente, atingindo a menor latência média (1,65 s) e a maior estabilidade, com desvio padrão abaixo de 0,5 segundos.

Observa-se que enquanto os métodos que fazem o *Range Request* mantém um *throughput* superior a 5 req/seg, a API não consegue atingir uma requisição completa por segundo. Essa diferença de mais de dez vezes na capacidade de processamento confirma

que o gargalo da API reside na latência de servidor do provedor. Diferente do que foi observado no cenário anterior em que o hardware e o tráfego de rede volumoso causavam grandes oscilações, a transferência seletiva se mostrou mais resiliente.

Para os métodos S3 e URL, o desempenho permaneceu estável em todos os períodos, com pequenas variações de tempo. Já a Web API apresentou seu pior desempenho no período da tarde, com o aumento da latência média (11,25 s) e o maior desvio padrão registrado (2,08 s), isso indica que a infraestrutura do provedor sofre maior impacto de concorrência nesse horário. Esses resultados validam que a descentralização da lógica de filtragem é a estratégia mais eficiente, corroborando com autores que identificaram os *Range Requests* via HTTP como alternativa intermediária eficiente entre APIs tradicionais e acesso direto a objetos em nuvem [Gallagher et al. 2024].

Apesar da eficiência desse cenário, observou-se novos desafios de confiabilidade durante a execução dos testes. A API apresentou erros do tipo *ERROR 102 USER QUEUED LIMIT EXCEEDED*, indicando que o provedor limita a fila de processamento a no máximo 20 requisições simultâneas por usuário. Em um cenário operacional real, essa restrição impede que essa etapa do pipeline processe múltiplos conjuntos de dados em paralelo. Em contrapartida, os métodos S3 e URL também apresentaram restrições com erros de camada de transporte como *Too Many Requests* para ambos e *Slow Down* para o *Bucket*. As médias de desempenho reportadas contemplam exclusivamente os testes bem sucedidos.

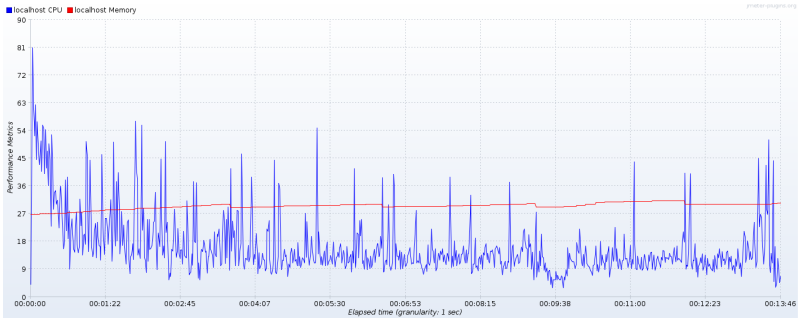

Figura 7. Consumo de CPU e Memória - método API Web Oficial (Manhã). A linha azul indica o consumo de CPU e a vermelha o consumo de memória.

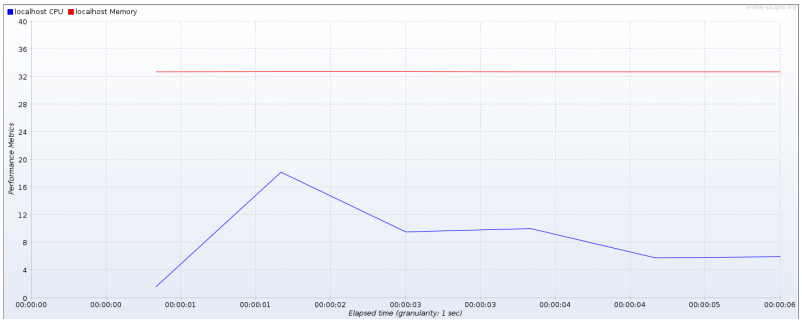

Figura 8. Consumo de CPU e Memória - método *Bucket* S3 (Noite). A linha azul indica o consumo de CPU e a vermelha o consumo de memória.

Além disso, identificou-se uma dependência operacional com o arquivo de índice. Para esse método o sistema realiza uma requisição inicial via URL direta para obter esse

arquivo, isso torna o *Bucket* S3 limitado pela janela de disponibilidade da URL, que são de quatro dias, impedindo que ele tenha independência na coleta, visto que o S3 possui uma maior disponibilidade em seus dados.

As figuras 7 e 8 apresentam o consumo de recursos no Cenário 2. O primeiro gráfico ilustra um caso atípico registrado durante a execução da API no período da manhã, onde a transferência se estendeu por mais de 13 minutos, com picos de CPU e oscilações contínuas durante todo o ciclo de *polling*. Esse perfil indica que o cliente permaneceu em espera prolongada pelo processamento remoto, consumindo recursos locais mesmo sem realizar transferência efetiva de dados. Em contraste, o segundo gráfico, referente ao *Bucket* S3 no período da noite, apresenta uma entrega concluída em menos de 6 segundos, com CPU abaixo de 20% e memória estável durante toda a execução. Essa diferença confirma que a descentralização da filtragem via arquivo de índice não apenas reduz o volume transferido, mas também preserva os recursos computacionais locais, tornando o método mais adequado para pipelines com atualizações frequentes.

5. Conclusão

Esse trabalho apresentou um estudo experimental comparativo entre três estratégias de coleta de dados meteorológicos do ECMWF. O objetivo foi identificar a abordagem mais eficiente e resiliente para integrar pipelines operacionais. Para a transferência integral, identificou-se que o hardware local e o tráfego de rede atuam como gargalos físicos, tornando o processo lento e sujeito a erros de conexão. A transferência seletiva, permitiu reduzir o volume de dados trafegados em 98%, reduzindo arquivos com tamanhos médios de 120MB em 2MB.

A análise revelou que a API é penalizada por uma latência de servidor inerente ao sistema de filas, o que restringe o escalonamento dessa etapa. Além disso, o monitoramento de recursos locais indicou que a coleta baseada em índices é mais estável e consome significativamente menos CPU e memória RAM do que a API, que se mostrou mais instável, preservando a integridade do hardware do processamento. A principal contribuição desse estudo foi a demonstração de que a descentralização da lógica de filtragem, utilizando arquivos de índice e *Range Requests*, é superior ao processamento centralizado das APIs. O método via HTTP direto destacou-se pela eficiência técnica, atingindo *throughput* até dez vezes superiores à Web API e garantindo maior resiliência para operações em tempo quase real.

Contudo, é reconhecido que este estudo experimental foi limitado a um ambiente de hardware local (HDD) e carece de uma análise estatística rigorosa para validar o desempenho obtido. Nesse cenário, as limitações de infraestrutura e oscilações de rede atuaram como gargalos, reforçando a necessidade de ambientes mais robustos para coletas volumosas. Sugere-se então, a adoção de uma arquitetura de transferência híbrida, que deve priorizar o acesso seletivo via URL direta para garantir a menor latência. Para trabalhos futuros, pretende-se realizar uma análise sistematizada em ambiente de nuvem, permitindo avaliar o comportamento dos protocolos sob escalabilidade e uma análise de custo para identificar, otimizar e prever os gastos financeiros e operacionais gerados pela transferência. Esse desdobramento focará na automação do fluxo de coleta e no monitoramento da latência de ponta a ponta, avaliando como a escolha do protocolo impacta as etapas subsequentes do pipeline meteorológico.

Referências

Apache Software Foundation (2026). Apache JMeter. `https://jmeter.apache.org/`. Acessado em: 16 de março de 2026.

Bauer, P., Thorpe, A., and Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55.

Bocchi, E., Mellia, M., and Sarni, S. (2014). Cloud storage service benchmarking: Methodologies and experimentations. In *Proc. 3rd IEEE International Conference on Cloud Networking (CloudNet)*, pages 395–400, Luxembourg.

ECMWF (2024a). Ecmwf web api documentation. `https://www.ecmwf.int/en/computing/software/ecmwf-web-api`. Acessado em: 14 de março de 2026.

ECMWF (2024b). Grib: Edition 2 - ecmwf parameter database. Acessado em: 13 mar. 2026.

ECMWF (2026a). About ecmwf. Acesso em: 10 fev. 2026.

ECMWF (2026b). Documentation and support: Changes to the ECMWF forecasting system. Acessado em: 23 mar. 2026.

ECMWF (2026c). Ecmwf open data. Acessado em: 13 mar. 2026.

ECMWF (2026d). S2S: Sub-seasonal to Seasonal Prediction Project. Acedido em: 17 mar. 2026.

ECMWF (2026e). TIGGE: Thorpex Interactive Grand Global Ensemble. Acedido em: 17 mar. 2026.

Fielding, R., Nottingham, M., and Reschke, J. (2022). HTTP semantics. RFC 9110. Acessado em: Acesso em: 10 jun. 2026.

Gallagher, J., Habermann, T., and Lee, J. (2024). Web accessible apis in the cloud trade study (task 28). Technical report, OPeNDAP / NASA.

Gowan, J., Habermann, T., and Jelenak, A. (2022a). Advancing cloud-native access to meteorological data. *Journal of Open Source Software*, 7(72).

Gowan, T. A., Horel, J. D., Jacques, A. A., et al. (2022b). Using cloud computing to analyze model output archived in Zarr format. *Journal of Atmospheric and Oceanic Technology*, 39(4):449–462.

Heise, E. (2019). GRIB2 for DUMMIES. COSMO Consortium General Meeting. Disponível em: `https://www.cosmo-model.org/content/consortium/generalMeetings/general2019/wg6/GRIB2-for-DUMMIES.pdf`. Acesso em: 10 jun. 2026.

Khemka, A. and Raj, G. (2025). Unstructured data ingestion: Best practices for acquiring, storing, and processing data from 200+ external sources. *International Journal of Research and Analytical Reviews (IJRAR)*, 12(1).

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

Pargaonkar, S. (2023). A comprehensive review of performance testing methodologies and best practices: Software quality engineering. *International Journal of Science and Research (IJSR)*, 12(8):2008–2014.

Rao, N. S. V., Liu, Q., Liu, Z., Kettimuthu, R., and Foster, I. (2019). Throughput analytics of data transfer infrastructures. In *Testbeds and Research Infrastructures for the Development of Networks and Communities*, volume 270 of *Lecture Notes in Computer Science*, pages 20–40. Springer.

Saeedizade, E., Zhang, B., and Arslan, E. (2023). Demystifying the performance of data transfers in High-Performance research networks. In *2023 IEEE 25th International Conference on High Performance Computing, Data, and Analytics*. IEEE.

Schwitalla, T., Warrach-Sagi, K., Wulfmeyer, V., and Bauer, H.-S. (2017). Continuous high-resolution midlatitude belt simulations for july-august 2013 with the wrf model. *Geoscientific Model Development*, 10(5):2031–2055.

WMO (2019). *Manual on Codes: International Codes, Volume I.2, Part B – Binary Codes*. World Meteorological Organization, Geneva, Switzerland. WMO-No. 306.