Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# BioCirrus: Uma Nova Política de Armazenamento para a Plataforma BioNimbuZ de Nuvem Federada

Diego Rodrigues Azevedo
Tarcísio Batista de Freitas Júnior

Brasília
2015

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# BioCirrus: Uma Nova Política de Armazenamento para a Plataforma BioNimbuZ de Nuvem Federada

Diego Rodrigues Azevedo
Tarcísio Batista de Freitas Júnior

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientadora
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo Von Paumgartten

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Computação — Licenciatura

Coordenador: Prof. Dr. Wilson Henrique Veneziano

Banca examinadora composta por:

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo Von Paumgartten (Orientadora) — CIC/UnB
Prof.ª Dr.ª María Emília Machado Telles Walter — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB

**CIP — Catalogação Internacional na Publicação**

Azevedo, Diego Rodrigues.

BioCirrus: Uma Nova Política de Armazenamento para a Plataforma BioNimbuZ de Nuvem Federada / Diego Rodrigues Azevedo, Tarcísio Batista de Freitas Júnior. Brasília : UnB, 2015.
121 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2015.

1. algoritmos de armazenamento, 2. nuvem federada, 3. compressão de arquivos, 4. estimativa de banda

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# BioCirrus: Uma Nova Política de Armazenamento para a Plataforma BioNimbuZ de Nuvem Federada

Diego Rodrigues Azevedo
Tarcísio Batista de Freitas Júnior

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo Von Paumgartten (Orientadora)
CIC/UnB

Prof.ª Dr.ª María Emília Machado Telles Walter
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
Coordenador do Curso de Computação — Licenciatura

Brasília, 30 de Agosto de 2015

# Dedicatória

Diego Azevedo: Dedico este trabalho aos meus pais, que me deram todas as condições necessárias à minha formação. Dedico também à minhas irmãs, que sempre quiseram meu melhor, mesmo que nossa definição de "melhor" fosse divergente. À minha namorada, que esteve do meu lado pelos últimos anos, aturando minhas chatices, e à nova alegria da minha vida, a Yuki.

Tarcísio Júnior: Dedico este trabalho à todos que estiveram ao meu lado, direta ou indiretamente durante meu percurso de graduação. Um agradecimento especial ao meus familiares e amigos, que me deram força e não me permitiram desistir. Um agradecimento também à minha namorada, que me ajudou a chegar longe me dando força sempre que necessário.

# Agradecimentos

Agradecemos primeiramente nossas famílias, responsáveis pela formação do nosso caráter e pelas condições que nos deram para superar esta etapa da vida. Devemos a eles quem somos hoje. Agradecemos também nossa orientadora, Prof.ª Dr.ª Aletéia Patrícia Favacho Araújo, por sua dedicação e principalmente à sua paciência (que não foi pouca) em nos ajudar neste projeto. Gostariamos de agradecer também ao Prof. MsC. Marcos Fagundes Caetano, pelo seu total apoio durante nossa graduação.

Não poderiamos deixar de agradecer a Prof.ª Dr.ª María Emília Machado Telles Walter e à Prof.ª Dr.ª Maristela Terto de Holanda pela gentileza de participar da nossa banca examinadora, cujas sugestões ajudaram a melhorar este trabalho.

Também agradecemos os colegas Breno Rodrigues Moura e Deric Lima Bacelar, pela sua contribuição no BioNimbuZ, base deste trabalho, e a todos os colegas que incentivam e contribuem para a plataforma, em especial ao Edward de Oliveira Ribeiro (MsC), um grande incentivador do projeto.

# Resumo

O foco deste trabalho é propor uma nova política de armazenamento para a plataforma de nuvem BioNimbuZ. Esta pesquisa é na plataforma de nuvem federada, que é um ambiente composto por provedores ligados entre si, e que compartilham recursos de uma maneira transparente ao usuário. Como a plataforma BioNimbuZ é voltada para aplicações de bioinformática, o tamanho dos arquivos nessa área alcança facilmente *gigabytes*, chegando à *terabytes* de dados. A nova política de armazenamento proposta neste trabalho, BioCirrus,leva em conta, além dos aspectos já existentes em trabalhos anteriores, a compactação e quebra de arquivos e o cálculo de banda entre cliente/servidor. O objetivo foi criar uma política com a transferência de arquivos mais rápida, com melhores decisões acerca do local de armazenamento que as políticas anteriores, e os resultados preliminares indicam que o objetivo foi alcançado.

**Palavras-chave:** algoritmos de armazenamento, nuvem federada, compressão de arquivos, estimativa de banda

# Abstract

This research's goal is to create and implement a better storage policy for a federated cloud platform, BioNimbuZ. It is focused in federated clouds, environments with many distinct cloud providers interconnected, sharing resources in the form of a single computational resource to the client. Since BioNimbuZ is focused in bioinformatics applications, the usual file size on the platform is very large, up to terabytes. This new policy, BioCirrus, uses bandwidth estimates, file compression, latency and uptime to make decisions where to send the file. The objective was to create a faster policy, with better decisions than the previous ones, and the preliminary results suggests this objective was achieved.

**Keywords:** Storage algorithms, federated cloud, file compression, bandwidth estimate

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

O mundo vem sofrendo diversas transformações desde o surgimento da Internet. Umas dessas modificações foi o uso do poder computacional, que aproveitou-se da rede mundial para ganhar em eficiência e escabilidade. Antes dessa evolução, a maior parte desse processamento era realizado em servidores locais e *datacenters*, que mesmo em seus momentos ociosos, geravam custos com manutenção e energia elétrica, com ou sem a utilização do seu poder computacional completo. Neste cenáario, surgiu a idéia de computação sob demanda, funcionando de maneira escalável para que os usuários paguem apenas pelo que forem utilizar.

Mesmo sabendo de sua funcionalidade, não existe na literatura uma definição correta para o termo computação em nuvem. Entretanto, é possível identificar alguns pontos chaves nestas definições, como ser um sistema distribuído, prover recursos por meio da Internet, trabalhar sob demanda, e ser escalável. Portanto, a computação em nuvem pode ser explicada como um sistema distribuído com o intuito de permitir o acesso à diversos recursos, como capacidade de processamento e armazenamento, que podem ser adquiridos como serviços. Assim, a computação em nuvem transmite a sensação de recursos ilimitados. Com o passar nos anos, novas necessidades surgiram e a utilização de nuvens separadas passou a ser insuficiente para suprir a demanda das aplicações pelo consumo de recursos [11].

Para solucionar este limite de recursos, nasceu o conceito de nuvens federadas, ou seja, um conjunto de nuvens interligadas, que funcionam de maneira transparente ao usuário, sem que este perceba que se trata de um grupo de nuvens. Portanto, em uma federação de nuvens, várias nuvens compartilham entre si seus recursos, permitindo uma capacidade de poder de processamento maior que a conseguida por meio de apenas uma nuvem [11].

Com o surgimento da federação em nuvem, muitas arquiteturas foram propostas, sempre com o intuito de ofertar seus recursos disponíveis de maneira a balancear suas tarefas. Uma dessas arquiteturas, chamada BioNimbuZ, foi proposta por Saldanha [32] em 2012.

Com o grande aumento do uso da computação em nuvem, principalmente para serviço de armazenamento, existe a busca por algoritmos eficientes de armazenamento. Como a plataforma BioNimbuZ trabalha com a execução de *workflows* de bioinformática, que comumente possuem grandes arquivos, é essencial para o bom desempenho do tempo de transferência da plataforma a eficiência do algoritmo de decisão de armazenamento, ou seja, a melhor escolha possível para o local de armazenamento. Esta pesquisa, portanto, propõe-se a aumentar a eficiência e melhoria do algoritmo de cálculo de armazenamento,

além de melhorar o uso da banda da federação, ao compactar grandes arquivos para transferência.

## 1.1 Problema

A política atual de armazenamento do BioNimbuZ não considera todos os pontos importantes da rede à qual está submetida para um cálculo eficiente de escolha de armazenamento dos arquivos. A nova política proposta tem como principal aspecto esta melhoria.

## 1.2 Objetivo

O objetivo principal deste trabalho é propor um algoritmo eficiente de decisão de armazenamento para a plataforma de nuvem federada BioNimbuZ. Para isso, é preciso cumprir os seguintes objetivos específicos:

- Propor um método eficiente para armazenamento de dado no BioNimbuZ;
- Implementar uma política de armazenamento eficiente;
- Testar em um ambiente real a política proposta.
- Analisar o desempenho do algoritmo proposto em uma estrutura real de nuvem federada.

## 1.3 Organização do Documento

No Capítulo 2 serão apresentados os conceitos de computação em nuvem, sua arquitetura, e suas possíveis classificações. Será também apresentado o conceito de federação em nuvens, e uma explicação sobre como funciona seu armazenamento.

O Capítulo 3 se destina a explanar o funcionamento da plataforma BioNimbuZ, e em especial suas características. No Capítulo 4 são apresentadas as fundamentações teóricas em que a nova política se baseia. Nessas fundamentações são analisados os conceitos de banda, assim como os princípios da compactação e da fragmentação de arquivos.

Os resultados obtidos a partir da nova política de armazenamento de dados são apresentados no Capítulo 5, no qual é feita uma comparação entre a nova política proposta (BioCirrus), e as políticas ZooClauS (atualmente implementada no BioNimbuZ) e a política proposta por Gallon [17].

Por fim, no Capítulo 6, são apresentadas as considerações finais sobre a política de armazenamento de dados proposta nesta pesquisa, e alguns trabalhos futuros que podem ser implementados.

# Capítulo 2

# Computação em Nuvem

O objetivo deste capítulo é apresentar, de maneira sucinta, a computação em nuvem, suas caracterítiscas e funcionalidades. Portanto, vários conceitos serão apresentados, sendo dado ênfase na apresentação do serviço de armazenamento de dados (*DaaS*). O entendimento desses conceitos é fundamental para a compreensão deste trabalho, que propõe uma nova política de armazenamento para a plataforma BioNimbuZ [32].

## 2.1 Histórico

Na década de 60 os computadores entraram na sua terceira geração, caracterizada pelo uso de transístores. Isso aumentou o processamento de dados, diminuiu o tamanho dos computadores, e otimizou o gasto elétrico [38]. Com essa evolução, ainda na década de 60, John McCarthy [22] propôs utilizar a computação de maneira pública, possibilitando que os recursos computacionais fossem utilizados sob demanda.

Mesmo com a melhoria alcançada pelo uso de transístores, os problemas acadêmicos continuaram se tornando mais complexos, demandando processamento computacional cada vez mais alto, até o ponto que o poder computacional disponibilizado por apenas uma máquina não era suficiente para atender a demanda de processamento de uma aplicação. Em decorrência disso, já na década de 70, surgiram os primeiros sistemas distribuídos, batizados de *clusters* [34]. O *cluster* é um conjunto de computadores em rede executando um sistema operacional específico, o que permite que as tarefas sejam processadas de maneira distribuída. Desta forma, em um *cluster* o conjunto de computadores é visto como um único recurso computacional.

Posteriormente, surgiram os *grids*, plataformas com algumas caracteríticas distintas dos *clusters*, como descentralização física dos nós e heterogeneidade em sistemas computacionais, entre outros. Nos *grids*, cabe ao cliente selecionar, agregar ou compartilhar os recursos escolhidos, baseado na diponibilidade, no custo e na performance desejados [8]. Apesar destas diferenças, as duas plataformas possuem recursos autônomos e garantem a imagem única de sistema para o usuário.

No começo do século XXI, algumas empresas passaram a oferecer software, plataforma e infraestrutura como serviços. Neste modelo, batizado de nuvem computacional, o cliente não precisa se preocupar com detalhes de infraestrutura, requisitando recursos computacionais virtualizados como serviço [7]. A Amazon, por exemplo, lançou sua plataforma

*Amazon Web Service* [23], em que os clientes podem requisitar máquinas virtuais com as mais diversas configurações, pagando por ciclos de CPU e espaço de armazenamento.

Logo, um dos grandes pontos fortes da computação em nuvem foi a maneira simples e transparente de alocar recursos. Assim, um cliente não fica limitado ao uso da capacidade física da sua máquina, e pode, de acordo com suas necessidades, contratar provedores que lhe concedam mais capacidade computacional [15].

### 2.1.1 Características das Nuvens

A computação em nuvem trabalha sobre virtualização e computação distribuída [8], que pode englobar supercomputadores, *mainframes*, *clusters*, *grids*, entre outros tipos de infraestrutura física. Portanto, leva em consideração a abstração de uma tecnologia a ser tratada como serviço (*as a Service*). Logo, os usuários não necessitam ter conhecimento sobre como o serviço é implementado, ou qual hardware é utilizado. Todos os detalhes são abstraídos e o serviço é entregue como um recurso virtualizado [21]. Nesse sentido, seus principais características são:

- *Self-service*: Usuários podem requisitar recursos computacionais diretamente, sem a necessidade de intervenção ou ajuda de um administrador. Normalmente, estas solicitações são feitas através de interfaces web;
- *Pooling* de Recursos: Os recursos computacionais do provedor são organizados em um *pool* para servir a múltiplos usuários, facilitando os ajustes de demanda;
- Serviço Mensurado: Os recursos são monitorados, e o cliente será cobrado de acordo com o que utilizar;
- Elasticidade: Os recursos podem ser adquiridos de maneira rápida e elástica, por vezes automática, caso seja necessário escalar com o aumento da demanda, ou liberar se houver diminuição de demanda;
- Amplo Acesso: Recursos são disponibilizados através da Internet e podem ser acessados por meio de dispositivos heterogêneos.

### 2.1.2 Arquitetura das Nuvens

A arquitetura em nuvem é dividida em camadas, sendo cada uma responsável por detalhes na disponibilização dos recursos. As camadas podem ser independentes em seus gerenciamentos e monitoramentos, oferecendo portabilidade. Assim, evitam impacto em outras camadas ao adicionar recursos [36]. As camadas são mostradas na Figura 2.1.

A arquitura se inicia contendo a parte de infraestrutura do sistema, o que inclui *clusters*, *desktops*, *datacenters* ou outros recursos. É, portanto, a parte física das camadas, podendo haver recursos heterogêneos, que aumentam a facilidade de inclusão da capacidade computacional, caso se faça necessário.

A camada do meio (*middleware*) possui duas subcamadas. A mais baixa (*middleware* central) é responsável por implementar a parte lógica que opera sobre a camada física.


Figura 2.1: Arquitetura de Computação em Nuvem, adaptado de [41].

É nessa camada que se encontram as negociações de QoS (*Quality of Service*), gerenciamento do SLA (*Service level agreement*), serviços de cobrança, serviço para cálculo, gerenciamento da virtualização, entre outros.

A subcamada mais alta do *middleware* (*middleware* nível de usuário) é responsável por prover suporte para a construção de aplicações e, portanto, possui ferramentas e ambientes de desenvolvimento. Por possuírem mais complexidade em sua estrutura, como interfaces, recursos de programação concorrente, bibliotecas e linguaguens de programação, esses ambientes são utilizados por desenvolvedores de aplicação em nuvem.

Por fim, a camada de maior interesse dos usuários é a de aplicação da computação em nuvem, pois é nessa camada que os aplicativos são utilizados. Portanto, para o usuário final, a camada de aplicação é a responsável pela interação com os recursos. O único pré-requisito é a Internet.

Dessa maneira, a computação em nuvem oferece serviços que são classificados de acordo com a camada em que trabalham, veja na Figura 2.2. Existem algumas taxonomias para classificação destes serviços, sendo a de Rimal e Choi [30] uma das mais comuns:

- Software como Serviço (SaaS): O SaaS proporciona softwares com funções que estão disponíveis para os usuários através da Internet, acessados por meio do navegador do cliente. No SaaS a estrutura que fornece o software não é controlada pelo cliente, seja o sistema operacional em uso, a capacidade da rede ou outros fatores externos à aplicação. O máximo que lhe é permitido são algumas configurações específicas.

  O software, por estar no meio Web, pode ser acessado pelos usuários de qualquer lugar e horário, permitindo agregação entre as unidades de uma empresa ou serviços de software. Com isso, novos recursos podem ser incorporados às aplicações sem que os usuários percebam, deixando a atualização do sistema transparente. Através do SaaS alguns custos podem ser evitados, como licenciamento de produtos e

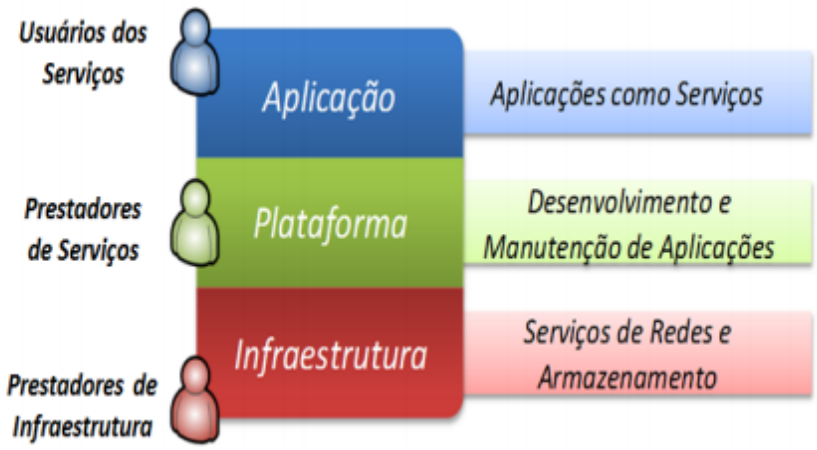


Figura 2.2: Os Atores e as Camadas de uma Plataforma em Nuvem, adaptado de [40].

manutenção de software. Exemplos de SaaS bastante utilizados são o Google Docs [19] e o Office 365 [25].

- Plataforma como Serviço (PaaS): O PaaS oferece infraestrutura para implementar e testar aplicações na nuvem. O cliente não possui controle sobre o nível mais baixo da estrutura, como rede, servidores, sistemas operacionais ou até armazenamento. A PaaS oferece estrutura pronta, como sistemas operacionais, linguagens de programação e ambientes de desenvolvimento para aplicação, auxiliando a implementação de softwares. Em geral, os desenvolvedores possuem ambientes escaláveis, mas são limitados por restrições sobre o tipo de software que pode ser desenvolvido. Essas limitações variam de concepções que o ambiente impõe, até o uso de valores de bancos de dados do tipo chave-valor. Do ponto de vista de negócio, a PaaS permite o uso de serviços de terceiros, aumentando o uso do modelo de suporte onde os usuários solicitam atendimentos de TI ou resoluções de problemas de rede. Exemplos de PaaS são o Google App Engine [18] e o Aneka [24].

- Infraestrutura como Serviço (IaaS): O IaaS é responsável por ceder os mecanismos físicos para SaaS e PaaS. Assim, seu objetivo principal é facilitar o fornecimento de recursos, tais como servidores, rede, capacidade de armazanenamento e quaisquer outros necessários para construir um ambiente de aplicação. Utiliza interface única para controle de administração da infraestrutura, API para comunicação com *hosts*, roteadores, *switches* e outros equipamentos de forma simples.

  O usuário, geralmente, não administra ou controla a infraestrutura alocada, mas pode gerenciar os sistemas operacionais, armazenamento e aplicativos implantados e, eventualmente, selecionar componentes de rede, tais como *firewalls*. O termo IaaS é relacionado a capacidade de virtualização dos recursos, maleável de acordo com o aumento ou diminuição da demanda. Exemplos de nuvens que oferecem IaaS são a Amazon EC2 (*Elastic Cloud Computing*) [23] e o Eucalyptus (*Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*) [14].

Entre os serviços disponibilizados pelo IaaS, existe o Serviço de Armazenamento de Dados [1]. Esse serviço é conhecido por DaaS (*Data Storage as a Service*) e oferece aos usuários uma capacidade de armazenamento expansível, a qual pode utilizar ou não banco de dados [10]. Dada sua importância para este trabalho, o DaaS será melhor explicado na Seção 2.3.

Além do tipo de serviço, as nuvens podem ser classificadas de acordo com sua implantação. A próxima seção explica essa classificação.

### 2.1.3 Classificação das Nuvens

Quando se trata do meio de acesso a ambientes em nuvem, existem algumas classificações possíveis. O tipo de nuvem que será utilizada depende da necessidade do usuário, que precisa levar em consideração o processo de negócio, o tipo de informação e a visão desejada. Dessa forma, as nuvens podem ser classificadas em [36]:

- Privada: As nuvens privadas são de exclusividade de uma empresa. Portanto, esta infraestrutura está em poder de seu contratante, que possui total controle sobre os aplicativos que serão implementandos. Comumente, os usuários precisam manter os softwares de infraestrutura da nuvem, o que implica em um maior custo. Se o usuário necessitar de maior poder computacional em algum requisito, ele precisará adquirir outro servidor, pois o seu é limitado à sua capacidade física. As nuvens privadas normalmente são utilizadas quando os dados são sigilosos, e portanto devem ser restritos por questão de segurança;

- Pública: Nas nuvens públicas a infraestrutura é disponibilizada para todos os usuários. Assim, basta conhecer a localização do serviço que está sendo provido. Neste modelo a segurança é baixa, já que não podem ser utilizados modos de restrição. Em contrapartida, possuem uma alta escalabilidade, possibilitando o acesso aos recursos extras quando necessário;

- Comunitária: Neste modelo ocorre o compartilhamento de uma nuvem por diversas empresas. Os usuários têm os mesmos interesses e, assim, irão estabelecer pontos como os requisitos de segurança, a política aplicada, etc [9]. Este tipo de modelo pode existir localmente, em que a empresa concede poder computacional de sua infraestrutura, ou remotamente. Via de regra, a própria empresa administra seu uso, em alguns poucos casos, deixa na responsabilidade para os usuários da comunidade;

- Híbrida: No modelo híbrido há uma mistura de nuvens públicas e privadas, mas que trabalham como entidade única, ligadas por uma tecnologia padronizada. Nele, os recursos de uma nuvem privada podem ser estendidos ao se utilizar recursos de nuvens públicas. Desta forma, há a vantagem da ampliação dos recursos disponíveis, porém é necessário possuir políticas de segurança, evitando que dados sigilosos saiam da organização.

Visto um pequeno histórico da computação em nuvem, assim como as formas que são implantadas e quais seus tipos, é importante conhecer o modelo de compartilhamento de recursos entre nuvens distintas. Assim, na Seção 2.2 será abordado o conceito de federação de nuvens.

## 2.2 Federação de Nuvens

Ao fornecer infraestrutura computacional como um serviço, com a possibilidade de se demandar mais recursos a qualquer momento, as nuvens criam a ilusão de que seus recursos são ilimitados. Mas, a limitação do hardware usado pelo provedor de serviços em nuvem existe, e pode haver picos de uso em uma plataforma que ultrapassa seu limite real. Para solucionar este problema, foi proposto o conceito de federação de nuvens [42].

O conceito de federação de nuvens implica no compartilhamento de recursos entre nuvens computacionais distintas. É um conceito bem amplo, usado quando um mesmo provedor possui diversas nuvens independentes (a Amazon, por exemplo, possui centros de dados independentes na Costa Leste, Oeste, Europa, Ásia, Oceania e América do Sul), ou quando nuvens de provedores distintos são integradas para garantir Qualidade de Serviço ($QoS$).

Segundo Buyya [8], como os provedores de serviços em nuvem não possuem *data centers* em todos os lugares do mundo, é difícil que consigam atender as expectativas de Qualidade de Serviço ($QoS$) de todos os seus consumidores. Por isso, é necessário construir mecanismos que possibilitem a federação de diferentes provedores de nuvens com o objetivo de atender metas de qualidade.

Neste cenário, a federação é a forma de integrar os ambientes de computação em nuvem, controlando e balanceando a demanda computacional. Para isso, é necessário que os provedores de nuvem disponibilizem seus recursos entre si, de maneira temporária ou permanente, dependendo do contrato estabelecido entre as nuvens. Isso também diminui o desperdício, pois os recursos ociosos de uma nuvem podem ser utilizados por outras.

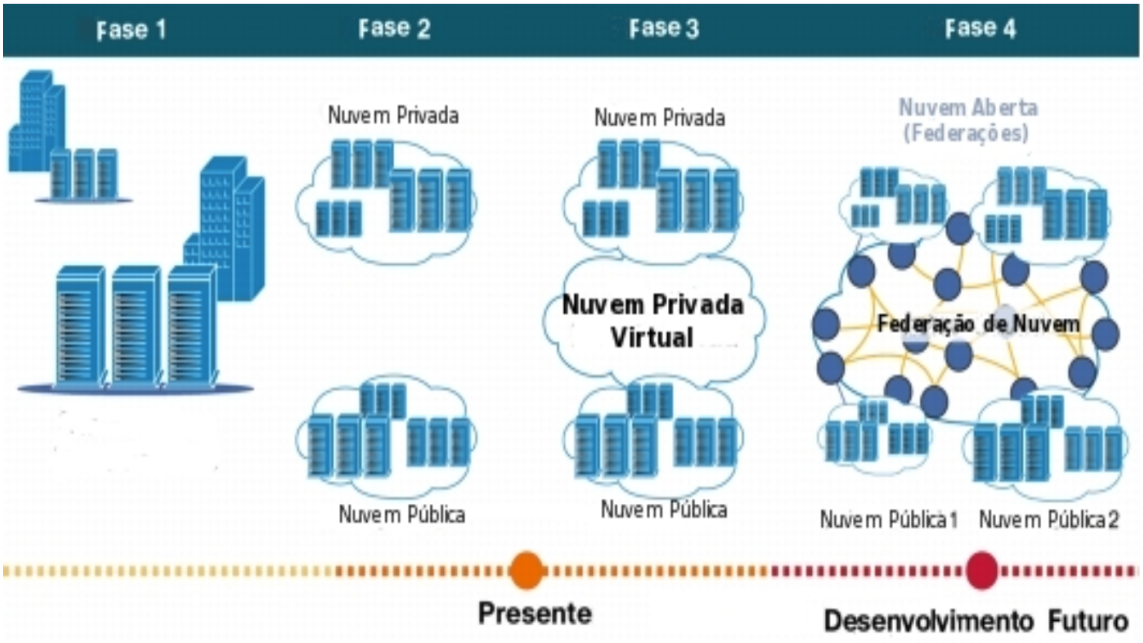


Figura 2.3: Fases da Computação em Nuvem, adaptado de [42].

White et. al [42] dividem o conceito de computação em nuvem em quatro fases, sendo duas já ultrapassadas, e as demais sendo trabalhadas e padronizadas, como é mostrado na Figura 2.3.

Na primeira fase, aparecem os *data centers* trabalhando de forma isolada, usando apenas a infraestrutura local. Não ocorre qualquer tipo de cooperação com outros *data centers*.

Na segunda fase, surge a estrutura das nuvens como é conhecida atualmente. Diversos *data centers* de uma mesma organização fornecem seus serviços por meio de nuvens computacionais. Nesta fase, ainda não há a interação entre nuvens de provedores distintos.

Na fase três, um provedor virtual começa a fazer parte do ambiente. É responsável por permitir a troca de recursos e o balanceamento de carga entre as nuvens, de modo a evitar a ociocidade de uma nuvem em um momento que outra se encontre com seus recursos em larga utilização.

Na quarta e última fase, as federações são espalhadas geograficamente, e trabalham de forma interdependente, dividindo seus recursos com outras federações da mesma forma que as nuvens dividem seus recursos dentro de uma federação.

Com a evolução da computação em nuvem, os serviços providos começaram a ser difundidos e utilizados com maior recorrência. Um dos serviços mais prestados é o de armazenamento, que será melhor explicado na Seção 2.3.

## 2.3 Armazenamento em Nuvens

Desde o surgimento da computação em nuvem, um de seus principais usos é o armazenamento de dados. Em nuvem, os dados são replicados em vários servidores, aumentando a segurança contra falhas de hardware ou exclusões acidentais. Isso permite o mecanismo de *failover*, no qual caso uma das máquinas fique *off-line*, os dados replicados em outras máquinas assumem essa atividade, tornando esse sistema bastante confiável e seguro [43]. Apesar da nuvem gerenciar o local de armazenamento dos dados dinamicamente, o usuário vê a localização dos dados de maneira estática.

Os recursos em nuvens, em geral, são financeiramente mais vantajosos do que os físicos, seja em computadores pessoais ou em rede [43]. O custo de manutenção de uma estrutura própria para armazenamento de dados tem aumentado bastante, tanto financeiro quanto fisicamente. Assim, as empresas vêm buscando alternativas mais viáveis. Em vez de gerenciar uma estrutura proprietária e dispendiosa, com altíssimo custo de manutenção, compram estes serviços sob demanda, na forma de armazenamento virtualizado, chamados de *Armazenamento de Dados como Serviço (DaaS)*.

Atualmente, não há uma padronização nos serviços de armazenamento de nuvem. A arquitetura dos serviços de armazenamento evoluiu de maneira heterogênea, como mostrado na Figura 2.4. Existem diversos protocolos de armazenamento, assim como arquiteturas consideravelmente diferentes, desde arquivos tradicionais (NFS) e bancos de dados SQL em *clusters* locais, até serviços na nuvem (por exemplo, a Amazon possui o S3 [3], EBS [5], SimpleDB [4], RDS [6] e ElastiCache [2]) [31].

Um modelo de serviço capaz de explicar e definir de forma geral o Armazenamento de Dados em Nuvem é o *Armazenamento de Dados Como Serviço (Data-storage-as-a-Service - DaaS)*. Devido ao seu uso em grande escala, tem grande relevância na computação distribuída, sendo um dos serviços ofertados pelo BioNimbuZ.

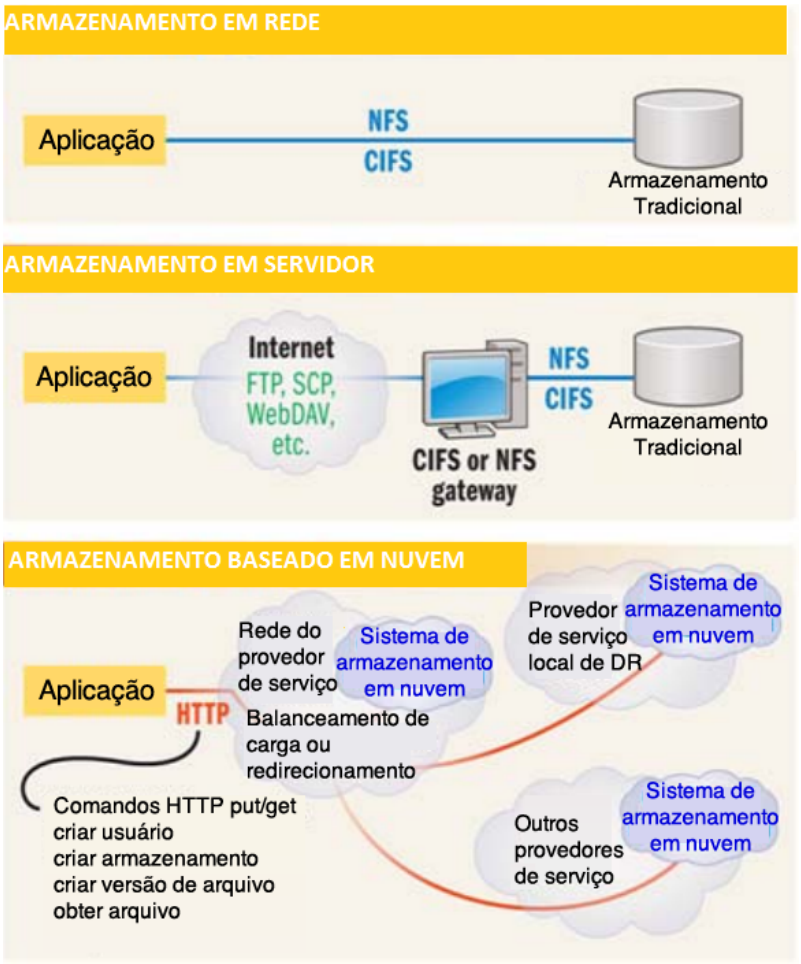


Figura 2.4: Evolução de Serviços de Armazenamento, adaptado de [43].

Quando se abstrai o armazenamento por um conjunto de ferramentas de serviços, entregues sob demanda, é possível ofertar um grande conjunto de aplicações para nuvem, exceto aquelas com quantidades fixas de recursos, como aplicações dedicadas. O único tipo que é excluído dessa definição é aquele que, em vez de ser entregue sob demanda, é entregue com capacidade fixa de recursos, aplicações dedicadas. Ao comprar uma aplicação dedicada, o cliente paga por recursos que foram alocados, mas não necessariamente são utilizados naquele momento. Na entrega sob demanda, a capacidade de armazenamento cresce à medida em que crescem os dados gerados pela aplicação [35].

A gestão dos dados armazenados na nuvem ocorre de diversas formas, e depende de como as ferramentas serão disponibilizadas pelo provedor do serviço. O modo mais comum é o gerenciamento remoto, que permite aos administradores que se conectem ao controlador de gerenciamento, quando, por alguma razão, estiver inativo, hibernando, ou não respondendo ao sistema operacional.

Deste conjunto de ofertas surgem diversas formas de abstrair o armazenamento de dados e entregá-los sob demanda. Uma das formas em crescimento, atualmente, é denominada *Database-as-a-Service (DbaaS)*, assim chamada por permitir que o acesso aos

dados seja feito por meio da linguagem SQL, ofertando na verdade um banco de dados em nuvem, que se diferencia do *DaaS* justamente nessa restrição de modo de acesso aos dados.

O *DaaS* é um modelo de arquitetura que provê armazenamento digital da sua estrutura como serviço. É apresentada como uma maneira segura e eficiente de manter *backups* e tem como vantagem principal a redução de custos. Esses serviços funcionam com base no *SLA (Service Level Agreement)* [39] assinado pelo cliente, que permite que o provedor do recurso cobre por *gigabyte* armazenado e pela transfência de dados. Assim, é garantida a segurança caso algum arquivo seja corrompido ou perdido.

## 2.4 Considerações Finais

Neste capítulo foram apresentados os conceitos básicos de computação em nuvem, federação de nuvens e a evolução dos sistemas de armazenamento em nuvens. Com eles é possível notar a grande utilização da computação em nuvem, tendo amplo uso para tarefas complexas, incluindo pesquisas no meio científico, como o projeto SIMAP [27], usado para o cálculo de similaridades entre proteínas.

Continuando esta abordagem, será vista no Capítulo 3 a plataforma de federação de nuvens BioNimbuZ [32], sua arquitetura e organização lógica, com um enfoque no serviço de armazenamento, que é o foco deste trabalho.

# Capítulo 3

# BioNimbuZ

Neste capítulo será apresentada a plataforma de federação de nuvens BioNimbuZ [32]. Na Seção 3.1 é apresentada a visão geral da plataforma, sua organização lógica e seus serviço de armazenamento. Na Seção 3.2 é detalhado o Serviço de Armazenamento e na Seção 3.3, as Políticas de Armazenamento propostas para o BioNimbuZ.

## 3.1 Visão Geral

O BioNimbuZ foi proposto por Saldanha [32] como uma arquitetura para federação de nuvens focada em executar *workflows* de bioinformática, integrando diferentes provedores de infraestrutura de maneira transparente, flexível e tolerante a falhas. Para facilitar o uso, já possui ferramentas de bioinformática oferecidas como serviço e, como toda plataforma de federação de nuvens, permite que o cliente tenha a ilusão de que os recursos computacionais são ilimitados, e que as demandas serão sempre atendidas. Para atingir esses objetivos, fez-se necessário integrar diferentes plataformas de computação em nuvem, com provedores diferentes, de maneira transparente.

Em 2013, o BioNimbuZ foi redefinido por Moura e Bacelar [26], e Oliveira [28] de forma a aumentar sua escalabilidade, sempre mantendo os princípios originais do BioNimbuZ, que são arquitetura flexível, modularidade e tolerância a falhas.

### 3.1.1 Arquitetura do BioNimbuZ

O BioNimbuZ possui uma arquitetura em componentes com funcionalidades bem definidas e divididas em três camadas: Camada de Aplicação, Camada de Núcleo e Camada de Infraestrutura (veja a Figura 3.1).

Na **Camada de Aplicação** é feita a comunicação com o usuário. É nesta camada em que são repassadas as requisições da aplicação e são recebidos os resultados após a execução. Esta interação pode ser feita a partir de páginas web, linhas de comando ou interface gráfica (GUI).

A **Camada de Núcleo** é a responsável pela gerência da federação, atualizando os recursos disponíveis, definindo o escalonamento de tarefas, o armazenamento de arquivos, o controle de acesso, entre outras atividades.

A **Camada de Infraestrutura** é a que consiste dos recursos que os provedores colocam à disposição da federação, junto de seus respectivos *plugins* de integração. A

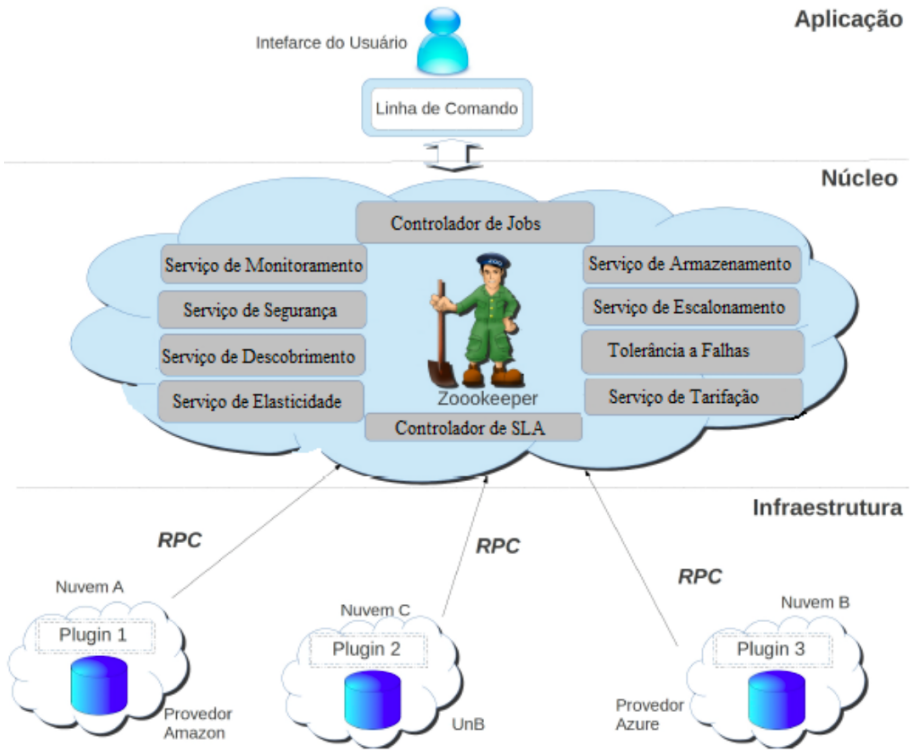


Figura 3.1: Arquitetura do BioNimbuZ [26].

integração do BioNimbuZ com diversos provedores de nuvens é feita por meio de um *plugin* de integração que implementa a interface de comunicação, abstraindo os detalhes do provedor.

Esse *plugin* mapeia as requisições no formato do BioNimbuZ para o formato específico do provedor de nuvem utilizado (e vice-versa), tornando a arquitetura flexível para a integração dos mais diversos provedores de nuvem computacional. O *plugin* também é responsável por coletar informações sobre os recursos computacionais disponíveis em seu provedor, bem como quais ferramentas de bioinformática são oferecidas. Estas informações são repassadas para os serviços controladores, no núcleo do BioNimbuZ.

Esses serviços trocam informações entre si, por meio de Chamadas de Procedimento Remoto (RPC), providas pelo Zookeeper [16], o qual é um serviço de coordenação para aplicações distribuídas. Cada componente envia um par de mensagens de requisição *Request* e resposta *Reply*. Os detalhes de cada um dos serviços da Camada de Núcleo estão descritos abaixo:

- ***Serviço de Descobrimento*:** Serviço responsável por coletar informações sobre as diversas nuvens computacionais federadas à plataforma, tais como recursos disponíveis e ferramentas oferecidas como serviço;
- ***Serviço de Escalonamento*:** Este serviço deve escalonar, iniciar, acompanhar e finalizar a execução das tarefas;
- ***Serviço de Armazenamento*:** Toda a estratégia de armazenamento dos arquivos consumidos e gerados pelas ferramentas executadas é feita por este serviço. Ele

também é responsável por coordenar estratégias de distribuição, replicação e controle de acesso dos arquivos entre os diferentes usuários e serviços da plataforma. Diante da importância deste serviço para este trabalho, o mesmo será detalhado na Seção 3.2;

- ***Tolerância a Falhas*:** Garante a disponibilidade dos principais serviços para que, em caso de falhas, o sistema possa ser recuperado;
- ***Serviço de Monitoramento*:** Realiza um acompanhamento dos recursos computacionais disponíveis junto ao Zookeeper [16], repassando estas informações para os outros serviços;
- ***Serviço de Segurança*:** A responsabilidade de gerenciar a política de acesso ao BioNimbuZ e às credenciais de acesso dos usuários a cada uma das nuvens da federação é deste serviço;
- ***Serviço de Elasticidade*:** Responsável por controlar dinamicamente o número de instâncias, bem como os parâmetros de uso dos recursos computacionais de cada uma dessas instâncias;
- ***Serviço de Tarifação*:** Responsável por calcular quanto os usuários devem pagar pelo uso dos recursos a partir do BioNimbuZ;
- ***Controlador de Jobs*:** A ligação entre a camada de núcleo e a de aplicação é feita por este módulo. Ele também verifica junto ao serviço de segurança as credenciais dos usuários, permitindo ou bloqueando a execução de tarefas;
- ***Controlador de SLA*:** É o módulo responsável por garantir o acordo de nível de serviço (*Service Level Agreement*), que garante a qualidade dos serviços prestados. Ao usar a federação, o usuário informa parâmetros de QoS (*Quality of Service*), e este serviço verifica se os requisitos do usuário são suportados pela federação.

Esses serviços são integrados por meio de uma interface bem definida. Desta forma, a mudança de implementação é uma tarefa simples, pois as funções e sinalizações utilizadas são as mesmas. Este modelo também permite que novas funcionalidades sejam implementadas na interface, tornado a arquitetura bastante extensível.

Além da organização interna, separada em camadas e serviços, o BioNimbuZ também possui uma organização lógica hierárquica, responsável por administrar os recursos computacionais da federação, que será apresentada na próxima seção.

### 3.1.2 Organização Lógica

Durante a evolução da plataforma BioNimbuZ, foram levantados diversos requisitos de eficiência e disponibilidade, que resultaram em uma organização hierárquica distribuída, como ilustrado na Figura 3.2. Assim, a hierarquia é dividida em três níveis, os quais são: *master* global, *master* local e *slave*. As principais características de cada nível são:

- ***Master* Global:** é o responsável por tomar as decisões de escalonamento, repassando-as para os demais recursos da federação. É ele quem ordena a execução de tarefas

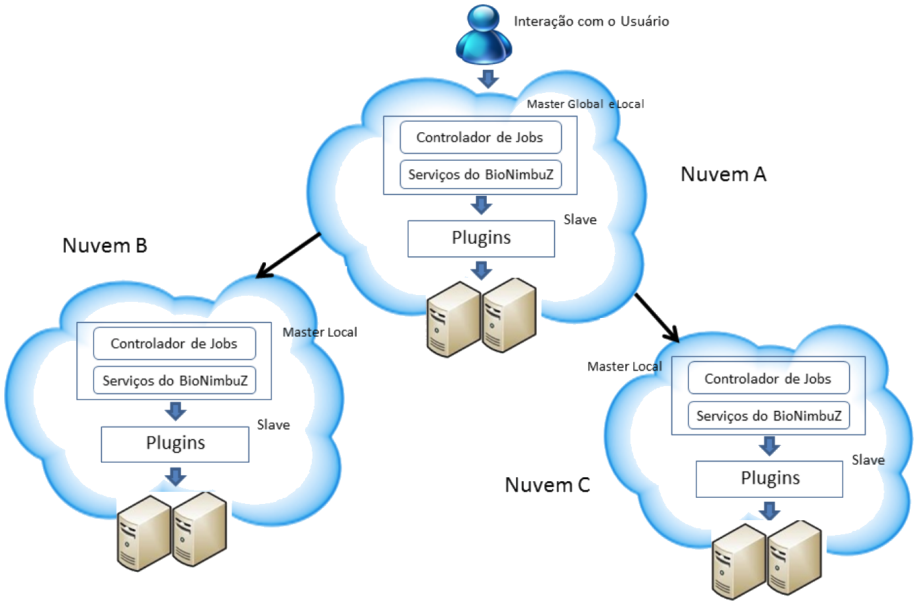


Figura 3.2: Organização Lógica do BioNimbuZ, adaptado de [13].

em outros provedores, através dos *masters* locais, recebendo deles o resultado final da execução. Só pode existir um *master* global ativo na federação;

- ***Master* Local:** recebe os pedidos de execução por parte do *master* global, repassando-as para os *slaves* para que sejam executadas. O *master* local também tem a responsabilidade de realizar o escalonamento local, informando ao *master* global o resultado da execução. Em cada provedor de nuvem existe um *master* local;
- ***Slave*:** executa as tarefas recebidas a partir do *master* local, repassando-lhe os resultados das execuções. Cada instância de máquina virtual do BioNimbuZ executa um *slave*.

O usuário, no entanto, não percebe essa divisão hierárquica, pois entra em contato apenas com o *master* global, como é ilustrado na Figura 3.2.

Um dos principais serviços do BioNimbuZ [32], bem como da maioria das plataformas em nuvem, é o de armazenamento de dados. Como este trabalho propõe uma nova política de armazenamento para a plataforma BioNimbuZ, este serviço será descrito na próxima seção.

## 3.2 Serviço de Armazenamento no BioNimbuZ

O serviço de armazenamento é o responsável por definir a estratégia usada para o armazenamento de arquivos que serão consumidos e que são entregues como *feedback* pela arquitetura BioNimbuZ, escolhendo os servidores para os quais os arquivos serão enviados e replicados.

Para realizar tal função, o serviço de armazenamento precisa de informações referentes às maquinas da federação. Para isso, envia mensagens do tipo *CloudReq* ao Serviço de

Descobrimento, para obter dados sobre as condições atuais de armazenamento de cada um dos provedores que fazem parte da federação.

Quando é feito um pedido de armazenamento de arquivo (*upload*), o local é escolhido pela interface *PluginInfo chooseStorage(FileInfo file)*. Uma estratégia de armazenamento deve implementar essa interface de forma a receber as informações de infraestrutura e retornar o recurso computacional escolhido para se armazenar o arquivo. Essa escolha é feita por um algoritmo que decide a partir dos dados recebidos pelo serviço de monitoramento.

Este serviço também é responsável por manter uma tabela com os arquivos armazenados na federação, bem como seus identificadores e localização. Esta tabela é chamada de *FileTable* e deve ser persistida com cópias em diversas máquinas (redundância), devido à criticidade desta informação para a federação.

Após cada *upload* bem-sucedido, confirmado por um *plugin* de integração, o serviço de armazenamento percorre todas as suas *FileTables*, acionando a interface *void storeFile* em todas elas. Assim, todas as tabelas são sincronizadas com o identificador dos arquivos, suas informações originais e seu local de armazenamento. Para que isto ocorra, o serviço de armazenamento possui três grupos de mensagens que deverá atender. O primeiro grupo de mensagens trata do processo de decisão do local de armazenamento dos arquivos, e as mensagens usadas são:

- *StoreReq*: Requisição de escolha do local de armazenamento de um arquivo. Deve conter dados da classe *FileInfo* para que sejam usados pela estratégia de armazenamento;
- *StoreReply*: Resposta do serviço de armazenamento indicando o local de armazenamento para o computador que solicitou o armazenamento. Deve conter, além dos dados da requisição, dados da classe *PluginInfo*;
- *StoreAck*: Mensagem enviada por um *plugin* de integração após a finalização do *upload* de um arquivo á sua infraestrutura. Ela contém dados da classe *PluginFile-Info* que serão armazenados nas tabelas de arquivo do serviço de armazenamento.

O segundo grupo de mensagens é ligado às consultas dos arquivos armazenados na federação, e as mensagens desse grupo são:

- *ListReq*: Requisição de listagem de arquivos armazenados na federação de nuvens. Ela não contém nenhum conteúdo adicional;
- *ListReply*: Resposta do serviço de armazenamento com uma coleção de dados da classe *File-Info*, que representa a consolidação de todos os arquivos mantidos na federação naquele momento.

O último grupo de mensagens recebida pelo serviço de armazenamento está relacionada à indicação do local do armazenamento do arquivo, a partir do qual, posteriormente, estará disponível para *download*, e as mensagens que compõem esse grupo são:

- *GetReq*: Requisição sobre o local de onde pode ser feito o *download* de um dado arquivo. Deve conter, pelo menos, o identificador único do arquivo;

- *GetReply*: Resposta do serviço de armazenamento com dados da classe *PluginInfo* sobre o provedor, para onde deve ser enviada uma mensagem do tipo *FilePrepReq* para iniciar o processo de *download*.

O serviço de armazenamento do BioNimbuZ necessita de uma política para definir o local de armazenamentos dos arquivos, como será visto na próxima Seção.

## 3.3 Trabalhos Relacionados

A plataforma BioNimbuZ foi concebida sem uma política de armazenamento definida, de modo que as decisões acerca do local de armazenamento eram feitas aleatoriamente. Desta forma, foram propostas duas políticas de armazenamento para a plataforma até a data atual, uma delas por Gallon [17] e outra por Moura e Bacelar [26], as quais são descritas a seguir.

### 3.3.1 Política Ricardo Gallon

A política proposta por Gallon [17] leva em consideração alguns critérios para escolher a melhor nuvem para armazenamento. Esses critérios são:

- Latência: Medida do atraso de um provedor a outro. Possuir latência mais elevada indica atrasos mais longos. A latência nunca pode ser eliminada inteiramente, e é usada como uma medida do desempenho da rede;
- Custo de armazenamento: O custo de armazenamento servirá apenas como critério de desempate. Havendo nuvens que apresentem condições semelhantes para realizar a operação, será priorizada aquela que não envolva custo;
- Núcleos de processadores O maior número de núcleos de processadores na nuvem candidata possibilita a resposta mais rápida à solicitação (*download* do sistema de arquivos e compressão);
- Carga de trabalho: Este critério visa garantir que a solicitação será atendida assim que for recebida pela nuvem. Havendo sobrecarga, o *download* do sistema de arquivos e a sua compressão estará comprometida, visto que não haverá recursos para serem utilizados nestas operações.

Assim, a seleção de nuvem para armazenamento será a que possuir o melhor índice (valor mais baixo) da seguinte equação:

$$I = (\frac{l}{max(l)} \times wl) + (\frac{ca}{max(ca)} \times wca) + (\frac{ct}{100} \times wct) - (\frac{np}{max(np)} \times wnp) \qquad (3.1)$$

Onde $I$ é o índice para seleção da nuvem, $l$ é a latência de rede entre a origem e o destino e tem peso 0.5, $ca$ é o custo de armazenamento da nuvem candidata e tem peso 0.05, $ct$ é a carga de trabalho da nuvem candidata e tem peso 0.15, e $np$ é a quantidade

de núcleos de processadores na nuvem e tem peso 0.3. Os pesos são: *wl* - latência, *wca* - custo de armazenamento, *wct* - carga de trabalho, *wnp* - núcleos de processadores.

Assim, a política de armazenamento Ricardo Gallon leva em consideração alguns pontos interessantes, como a latência, para escolher os recursos mais próximos, e o custo de armazenamento para evitar que o custo final seja muito elevado. Além disso, se utiliza de compactação dos arquivos para diminuir o tempo de transferência. No entanto, diversos fatores levados em conta pela política devem ser revistos, tais como:

- A latência, que é uma medida bastante rudimentar da qualidade do enlace, sendo muito mais interessante aferir a largura de banda (*bandwidth*) entre as máquinas;
- A carga de trabalho e o número de núcleos de processamento são medidas usadas em uma tentativa de prever o local de processamento escolhido pela poítica de escalonamento. No entanto, isso foge do escopo de uma política de armazenamento, uma vez que faz com que os arquivos sejam replicados em recursos mais caros. Outra falha desta abordagem é supor que os arquivos serão utilizados em seguida, e nem sempre esta situação se confirma.

Estes foram pontos a serem revisados e alterados na nova política, dando importância a outros pontos mais relevantes, como mostrado no Capítulo 4.

### 3.3.2 Política ZooClouS

Em 2013, Moura e Bacelar [26] propuseram uma política de armazenamento para a plataforma BioNimbuZ, chamada ZooClouS. A política baseia-se principalmente no fato do BioNimbuZ ser usado para executar *workflows* de bioinformática, e seus arquivos resultantes terem as características ressaltadas:

- O enorme volume que normalmente ocupam; e
- Durante as execuções de ferramentas de bioinformática não há atualização simultânea de dados por mais de um usuário.

Desta forma, a política ZooClouS decide a localidade onde deve ser enviado um arquivo por meio de um algoritmo que calcula um custo de armazenamento para o arquivo, baseado nas seguintes variáveis:

- *Free-Size*: Espaço livre em disco do servidor;
- *Uptime*: Tempo que o servidor está *online*. Tem o segundo maior peso no cálculo de armazenamento, dada a importância da confiabilidade gerada pela possibilidade de desconexão de um servidor. Toda vez que um servidor é adicionado a federação, é adicionado um valor em milisegundos ao seu *znode*, para se ter conhecimento do tempo que este está *on-line*.
- *Latency*: A latência (atraso de envio de pacote) entre as máquinas da federação é a variável com maior peso no custo.

Para realizar o cálculo do custo de armazenamento em um determinado servidor, a política ZooClouS utiliza a seguinte equação:

$$S = \frac{(L \times \alpha)}{(U \times \gamma + F \times \beta)} + Costs$$

Onde $L$ é a latência (*latency*), $U$ é o tempo *on-line* (*uptime*), $F$ é o espaço disponível (*free-size*) e $Costs$ é o custo por *gigabyte* que algumas nuvens, como a Amazon [23] e e Azure [12] cobram para armazenar o arquivo. Os valores de $\alpha$, $\beta$ e $\gamma$, tal que $\alpha+\beta+\gamma = 1.0$ foram definidos empiricamente, sendo eles 0.5 para $\alpha$, referente à latência, 0.2 à constante $\beta$, relativa ao *freesize*, e 0.3 à constante $\gamma$, atribuída ao *uptime*.

Depois de calculados os valores de cada servidor da nuvem, uma lista ordenada de maneira crescente pelos seus custos de armazenamento, é recebida com todos os servidores disponíveis. Assim, considera-se que quanto menor o custo, melhor é a transferência de um arquivo para o servidor.

A política ZooClouS proposta por Moura e Bacelar [26] possui aspectos bastante interessantes, como tentar aferir a qualidade do enlace e a confiabilidade do servidor de destino. Porém, há diversas possibilidades de melhorias, tais como:

- A variável *Uptime*, medida em segundos pela política ZooClouS, cresce muito rapidamente, sem restrições. É interessante fazer essa medida em minutos ou horas, limitando o crescimento até um valor de confiabilidade máxima;
- A variável *Free Space* não agrega valor à política, uma vez que os servidores com capacidade para receber o arquivo são previamente selecionados;
- A latência, como visto na Seção 3.3.1, é uma medida mais rudimentar que a largura de banda (*bandwidth*) para aferir a qualidade da rede;
- Arquivos de bioinformática são facilmente compactados a uma boa taxa de compressão, uma vez que se tratam de arquivos de texto puro. Dessa forma, é interessante que sejam comprimidos antes de serem enviados, diminuindo assim o tempo de transferência e economizando recursos de rede.

Desse modo, o Capítulo 4 propõe uma nova política de armazenamento, capaz de suprir as deficiências da política ZooClouS.

## 3.4 Considerações Finais

Este capítulo abordou a plataforma BioNimbuZ de federação de nuvens, sua arquitetura e organização lógica. Também foi abordado, com mais profundidade, o serviço de armazenamento e a sua atual política, a ZooClouS, com suas vantagens e possibilidades de melhorias.

No próximo capítulo será apresentada uma nova política para a plataforma BioNimbuZ chamada BioCirrus, proposta neste trabalho, que é capaz de suprir as deficiências das políticas Ricardo Gallon e ZooClouS.

# Capítulo 4

# BioCirrus

Este capítulo apresenta a política de armazenamento proposta nesse trabalho, denominada BioCirrus. Na Seção 4.1 é descrita a política proposta detalhamente. Na Seção 4.2 são apresentados os métodos utilizados na nova política, os quais são o cáculo da largura de banda, a compressão dos arquivos, e a fragmentação dos arquivos. E na Seção 4.3 é apresentado o cálculo da escolha de armazenamento da política proposta.

## 4.1 Política de Armazenamento Proposta

A política BioCirrus se baseou nas políticas anteriores, descritas na Seção 3.3, e nos arquivos que são trafegados pela plataforma. Como a plataforma de nuvem federada BioNimbuZ é usada principalmente para executar *workflows* de bioinformática, seus arquivos têm as seguintes características:

- São arquivos bastante extensos, extrapolando facilmente os 200 *megabytes*, podendo chegar até a casa dos *terabytes*;
- São arquivos de texto puro, utilizando-se muitas vezes de apenas quatro caracteres: A, C, G e U, que representam as bases nitrogenadas Adenina, Citosina, Guanina e Uracila, respectivamente.

Devido essas características, foram levantadas as seguintes hipóteses para a melhoria do desempenho da plataforma:

- É vantajoso aferir com mais precisão a qualidade do enlace entre os diversos recursos da federação;
- Compactar os arquivos antes da transferência provoca diminuição do tempo total do envio;
- Dividir os arquivos antes de compactactá-los resulta em melhor tempo de envio.

No entanto, os estudos indicam que algumas características usadas pelas políticas de armazenamento implementadas anteriormentente no BioNimbuZ, a política ZooClous[26] e Ricardo Gallon[17], deveriam ser mantidas:

- A latência continuará sendo usada como um dos indicativos de qualidade do enlace, mas deixará de ser único, como em ambas as políticas citadas;
- O custo por *gigabyte* armazenado será mantido, por ser uma medida importante para diminuir o custo do uso da plataforma;
- O *uptime* continuará sendo usado como indicativo único da confiabilidade do servidor, mas deixará de ser uma medida linear.

As seguintes características das políticas deixarão de ser usadas, por serem consideradas desnecessárias ou fora do escopo de uma política de armazenamento:

- O *free space* deixará de interferir na fórmula, uma vez que os servidores com espaço suficiente em disco são filtrados previamente, de modo que todas as máquinas analisadas pela política de armazenamento já são consideradas aptas a receber o arquivo;
- O número de núcleos de processador também será descartado da fórmula, uma vez que foge ao escopo da política de armazenamento tomar decisões de escalonamento;
- A carga de trabalho da máquina também deixará de ser um item considerado pelo mesmo motivo, uma vez que esse é um critério de escalonamento, não armazenamento.

Na próxima seção, os aspectos escolhidos para se definir o local de armazenamento serão detalhados, juntamente com um estudo para se verificar a validade das hipóteses iniciais.

## 4.2 Variáveis da Política BioCirrus

A política BioCirrus, como apresentada na seção anterior, se baseia no tempo gasto com o tráfego de dados (qualidade do enlace), na divisão e na compactação dos arquivos e na confiabilidade dos recursos na rede. Desta forma, esta seção apresenta cada uma destas variáveis.

### 4.2.1 Qualidade do Enlace

Para validar a primeira hipótese "É vantajoso aferir com mais precisão a qualidade do enlace entre os diversos recursos da federação", é preciso compreender os dois principais conceitos que interferem no tempo de transferência, a latência e a largura de banda. Seu entendimento é fundamental para trabalhar com técnicas de performance e otimização em transferência de arquivos em redes.

A latência refere-se ao tempo que um sinal demora a chegar ao seu receptor. De modo geral, é a demora no envio de mensagens através de uma rede, ou através da Internet [37]. Quanto maior for o período de demora, maior a latência, que tem seu fator medido em milissegundos.

A largura de banda (*bandwidth*) é definida como a capacidade de transmissão de algum meio, e determina a velocidade com que os dados trafegam [29]. A largura de banda é, por padrão, medida em bits, e determinada sobre uma certa unidade de tempo. As medidas de largura de banda são, assim, calculadas em bits por segundo.

Quanto maior a largura, maior a capacidade de transferência de dados, seja de maneira paralela (envio de arquivos simultâneos) ou singular (envio de um único arquivo). Alguns fatores, tais como capacidade de servidor, tecnologia usada no caminho percorrido pelo dado e o número de usuários acessando o canal, influenciam diretamente a capacidade da taxa de transferência. Assim, a largura de banda é uma variável que deve ser calculada e atualizada sempre que necessário.

A latência e a largura de banda medem o tempo de resposta e a quantidade de dados trafegados em paralelo, respectivamente. Traçando uma analogia de uma rede com uma série de tubos interconectados, a latência seria a distância dos tubos entre dois pontos, e a largura de banda seria o quão largos os tubos são. Assim, quanto mais largos os tubos, maior e mais rápida é a capacidade de envio. Essa analogia é mostrada na Figura 4.1.

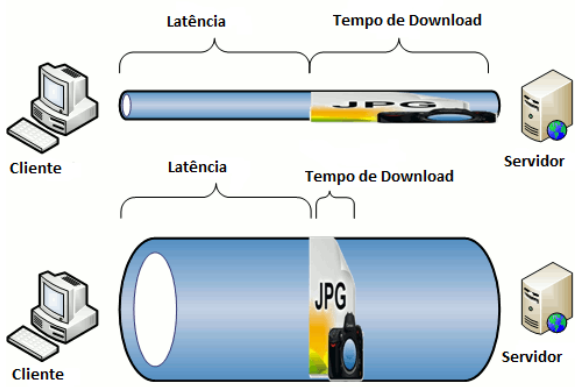


Figura 4.1: Latência e Largura de Banda, adaptado de Hoffman [20].

Na situação ilustrada na Figura 4.1, tem-se a simulação do envio de um arquivo de um servidor para o cliente com larguras de banda distintas, mas com a mesma latência. No caso ilustrado, é possível observar que a latência influencia bastante no tempo de transferência, uma vez que o tempo de *download* é muito pequeno.

A latência, como mostrado na Seção 3.3, já é levada em consideração nas políticas ZooClouS[26] e Ricardo Gallon [17] para escolha do local onde ocorrerá o armazenamento. Mas como o cálculo de banda também tem grande importância, ele será utilizado pela nova política. A importância da largura de banda pode ser analisada através da Equação 4.1 para cálculo de tempo de transferência:

$$time = \frac{size}{band} + lat \tag{4.1}$$

Onde *time* é o tempo para transferência de um arquivo, *size* é o tamanho do arquivo, *band* é a largura de banda, e *lat* é a latência. Assim, através dessa equação, e sabendo que arquivos de bioinformática são, em geral, muito grandes, nota-se que a primeira parte da equação ($\frac{size}{band}$), tem maior relevância para o tempo total, uma vez que o valor da latência gira em torno de milissegundos.

Dessa forma, sabendo que a largura de banda é responsável assintoticamente pelo tempo gasto na transferência, foi realizado um estudo para determinar a melhor maneira de estimá-la antes da transferência, o qual é apresentado a seguir.

### Largura de Banda no BioNimbuZ

Como foi mostrado, não se aconselha, do ponto de vista da qualidade da rede, levar em consideração apenas a latência, como é feito pela política ZooClous, dada a importância da largura de banda.

Para estimar a *bandwidth* entre os recursos, foi inserido um arquivo no servidor, o qual o cliente faz *download* ao requisitar serviço de armazenamento. Dessa forma, foi medido o tempo gasto para este *download*. Sabendo o tamanho do arquivo, é possível estimar a largura de banda, como mostrado na Equação 4.2:

$$band = \frac{size}{time - lat} \tag{4.2}$$

Onde *band* é o valor da largura de banda, *size* é o tamanho do arquivo transferido, *time* o tempo gasto para a transferência deste arquivo, e *lat* a latência entre as máquinas. O fator *lat* foi subtraído da taxa de transferência para um cáculo mais preciso da banda, que deve ser analisada sem o atraso do envio do arquivo. O último ponto analisado foi o tamanho ideal de arquivo, para que o cálculo se tornasse o mais preciso possível. Assim, por meio dos testes realizados foi possível obter a análise apresentada na Figura 4.2.

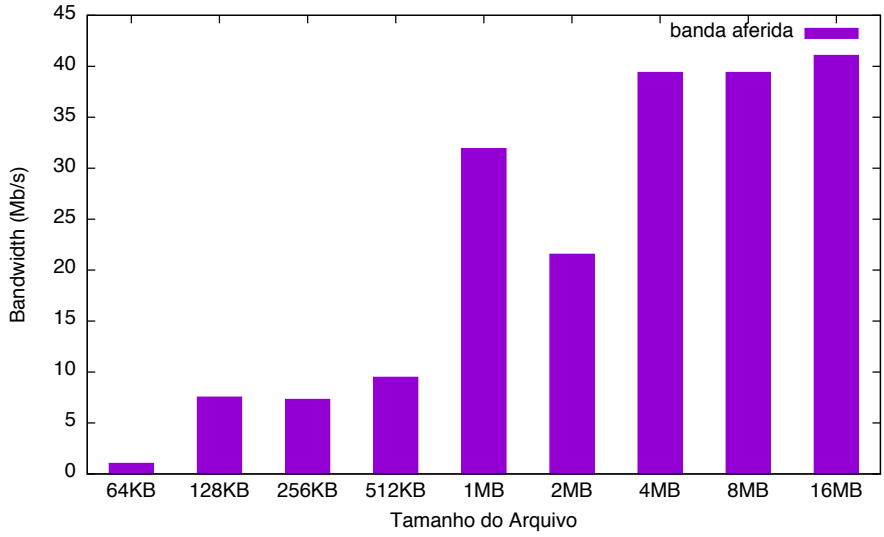


Figura 4.2: Precisão de Cálculo de Banda por Tamanho de Arquivo.

Na Figura 4.2 tem-se um gráfico de largura de banda (eixo y), por tamanho de arquivo (eixo x), para que seja aferida de maneira precisa a taxa de transferência. Por decorrência da oscilação de banda, o tempo para transferência de arquivos não se torna linear com arquivos de até 2 MB, causando uma variação nos valores obtidos. Como mostrado na Figura 4.2, transferências de arquivos de 4 MB já conseguem medir com mais segurança os valores de banda, pois demandam mais tempo para *download*, possibilitando um cálculo

de banda mais exato. Assim, foi definido como 4 MB o tamanho ideal de arquivo para se estimar a largura de banda entre recursos do BioNimbuZ.

Além da estimativa da qualidade do enlace, baseado na latência e na banda, a política BioCirrus se utiliza da compressão de arquivos para otimizar o tempo de transfência, apresentado na próxima seção.

### 4.2.2 Compactação de Arquivos

A segunda hipótese em que a política BioCirrus se baseia "Compactar os arquivos antes da transferência deve diminuir o tempo total do envio", parte da premissa de que o tempo gasto na compactação é menor de que o tempo ganho ao se fazer a transferência de um arquivo menor. Desta forma, nesta seção são apresentados os diferentes tipos de compressão e um estudo realizado para se validar a hipótese citada.

A compressão de dados é o termo usado ao se aplicar uma estratégia de diminuição de espaço ocupado por um determinado arquivo[33]. Estas estratégias são classificadas em *com perdas* e *sem perdas*, baseadas na capacidade de se restaurar o arquivo original após o processo de compressão.

A compressão com perdas (*lossy compression*) é usada principalmente em arquivos de mídia (fotos, imagens e vídeos), em que partes consideradas insignificantes dos dados são descartadas, o que reduz o tamanho total do arquivo, como também sua qualidade. Isso pode ser feito de diversas formas, como a diminuição das paletas de cores de uma imagem.

A compactação sem perdas (*lossless compression*) consiste em diminuir o espaço ocupado no disco ao realocar a informação de maneira mais inteligente, evitando redundâncias. Nesta forma de compressão de dados não há perda de informação, e o dado descompactado é idêntico ao original[33]. Por exemplo, a sequência AAAAAA, que ocupa 6 bytes, poderia ser subtituída por 6A, que ocupa apenas 2 bytes.

Devido ao fato do BioNimbuZ ser utilizado, principalmente, para executar *workflows* de bioinformática, com resultados que exigem altíssima precisão, não é aceitável que haja perda no processo de compressão. Desta forma, foram analizadas diversas ferramentas de compactação *lossless*. Para avaliar a compressão dos arquivos, bem como escolher o compactador ideal dependendo da taxa de transferência do arquivo na rede, foram pesquisadas diversas ferramentas existentes no mercado para a linguagem Java, e suas possíveis configurações. Logo, as ferramentas analisadas foram: Apache BZip, Apache Deflate, Apache GZip, Apache Snappy, Apache XZ, Java GZip, Java Zip, Zip4J (com 5 níveis de compressão), ZLib (com 3 estratégias: *Default*, *Filtered*, e *Huffman* e 5 níveis de compressão).

O estudo redigido neste trabalho baseou-se na premissa de que a soma dos tempos gastos na compressão e envio deve ser menor que o tempo de envio do arquivo sem compactação. Como visto na Seção 4.2.1, o tempo gasto na transferência é assintoticamente definido pelo tamanho do arquivo e largura de banda. Dessa forma, foi estimado o tempo de transferência de um arquivo de 1MB sobre as diversas formas de compactação, baseada na Equação 4.3.

$$t(x) = \frac{1 - ratio}{x} + time \tag{4.3}$$

Onde $t(x)$ é o tempo estimado de transferência com largura de banda $x$. A taxa de compactação é descrita por *ratio* e o tempo gasto na compressão, por *time*. Assim, foi possível calcular os pontos de interseção, e verificar o domínio em que cada compactador é ótimo. Na Figura 4.3 é demonstrada a relação entre tempo (eixo $y$), e largura de banda (eixo $x$). Logo, é inferido que, quando a taxa de transmissão é extremamente alta, sua transferência deva ser direta.

Por outro lado, as estratégias de compressão mais lentas (*time* mais alto), têm melhor taxa de compactação (*ratio*). Como é mostrado na Figura 4.4, que demonstra a relação de tempo (eixo $y$), com pequenas larguras de banda (eixo $x$), há um momento na curva, para velocidades de rede muito baixas, em que é melhor compactar o arquivo com uma estratégia mais lenta, de modo a ganhar tempo ao transmitir um arquivo ainda menor.

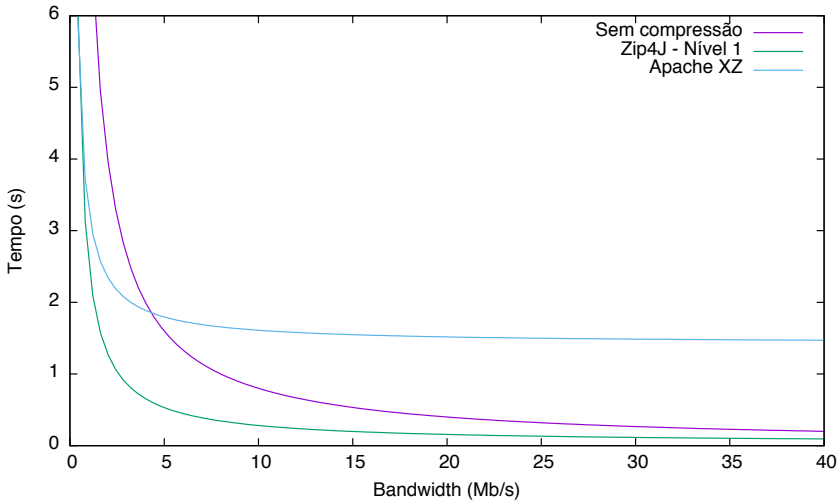


Figura 4.3: Curvas de Tempo Para Diferentes Compactadores.

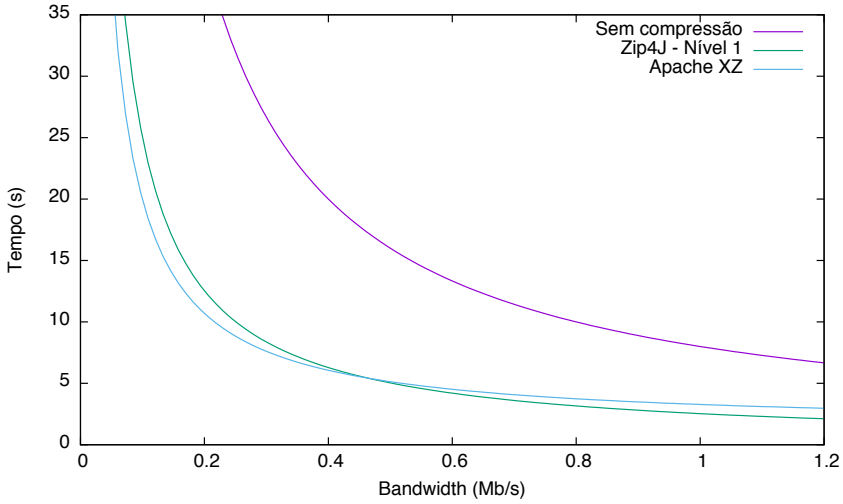


Figura 4.4: Interseção das Curvas Obtidas a partir da Figura 4.3.

Dessa forma, estimou-se que, mesmo para velocidades altíssimas de trasferência (até 295 Mb/s), há um ganho de tempo ao compactar o arquivo antes do envio. Porém, a escolha do compactador depende da largura de banda aferida.

Como a taxa e o tempo de compactação variam dependendo do tamanho do arquivo, na Seção 4.2.3 serão vistos os impactos da divisão de arquivos na compactação, e o limite de tamanho viável para quebra de arquivos utilizados no BioNimbuZ.

### 4.2.3 Divisão de Arquivos

A terceira hipótese utilizada pela política BioCirrus "Dividir os arquivos antes de compactactá-los resulta em melhor tempo de envio", baseia-se no fato da compressão de arquivos menores ser geralmente mais rápida, apesar de possuir uma menor taxa de compressão. Desta forma, nesta seção será apresentado um estudo verificando a relação divisão × compactação, com foco no tempo total de transferência.

Quando se trabalha com compactação de arquivos, é sabido que, quanto maior o tamanho, mais provável é a chance de redundâncias, e melhor é a taxa de compressão. No entanto, gasta-se mais tempo compactando um arquivo maior, o que resulta na necessidade de avaliar a relação entre a quebra de um arquivo, dependendo do seu tamanho.

Para descobrir qual é o melhor tamanho para dividir um arquivo antes da compactação, de modo a minimizar o tempo total (divisão, compactação e transferência), foi feito um estudo com todos os compactadores descritos na Seção 4.2.2, dividindo três arquivos reais de bioinformática, com aproximadamente 500 MB de tamanho total, analisando a relação entre tempo e taxa de compressão.

Os arquivos foram divididos em blocos de 1, 2, 4, 8, 16, 32, 64 e 128 MB, sendo cada parte compactada 20 vezes com cada uma das estratégias de compressão, possibilitando verificar o tempo (por MB) e a taxa média de compressão de cada compactador para cada tamanho de arquivo.

Com base nos dados obtidos e no fato da largura de banda ser o principal fator para o tempo de transferência (mostrado na Seção 4.2.1), foi definido um cálculo estimando a largura de banda máxima que deve ser interessante de utilizar em cada estratégia de compressão, para cada tamanho de arquivo. A largura de banda máxima foi obtida por meio da Equação 4.4, igualando o tempo de transferência de um arquivo sem compactação ($\frac{1}{band}$) e um arquivo compactado ($\frac{1-ratio}{band} + time$).

$$\frac{1}{band} = \frac{1 - ratio}{band} + time$$

$$\frac{1}{band} = \frac{(1 - ratio) + time \times band}{band}$$

$$1 = (1 - ratio) + times \times band$$

$$ratio = time \times band$$

$$\frac{ratio}{time} = band \tag{4.4}$$

Na equação 4.4 *band* é a largura de banda estimada, *ratio* é a taxa de compressão, e *time* é o tempo médio de compressão por *megabyte*. Dessa forma, foi possível analisar a relação entre a taxa e o tempo de compressão de maneira objetiva.

Para essa análise, os dados foram ordenados de forma decrescente pela relação $\frac{ratio}{time}$, como mostrado na Tabela 4.1. Nessa tabela são apresentadas quatro colunas, que mostram o compressor utilizado, o nível de compressão aplicado, a relação $\frac{ratio}{time}$, e por último, o tamanho do bloco, respectivamente. Assim, foi possível constatar que a compactação é mais vantajosa para arquivos maiores, invalidando a hipótese da divisão de arquivos trazer vantagens à compressão.

Tabela 4.1: Parte dos Resultados Obtidos a partir da Análise de Compressão.

| Compressor | Nível de Compressão | ratio/time | Tamanho |
|---|---|---|---|
| Snappy | - | 499.028 | 128 |
| Snappy | - | 404.790 | 64 |
| Snappy | - | 358.486 | 32 |
| Snappy | - | 351.151 | 16 |
| Snappy | - | 338.519 | 8 |
| Snappy | - | 329.391 | 4 |
| Snappy | - | 313.546 | 1 |
| Snappy | - | 312.511 | 2 |
| Zip4J | 1 | 289.021 | 128 |
| Zip4J | 2 | 234.069 | 128 |
| Zip4J | 1 | 229.827 | 64 |
| Zip4J | 1 | 199.624 | 32 |
| Zip4J | 2 | 186.954 | 64 |
| Zip4J | 1 | 186.102 | 8 |
| Zip4J | 1 | 185.637 | 16 |
| Zip4J | 1 | 182.792 | 4 |
| Zip4J | 4 | 180.218 | 128 |
| Zip4J | 1 | 179.515 | 2 |
| Zip4J | 1 | 173.577 | 1 |

O último fator levado em conta na política BioCirrus é a confiabilidade dos servidores disponíveis, apresentado na próxima seção.

### 4.2.4 Confiabilidade dos Recursos

Um dos mais importantes fatores a serem analisados na escolha do local de armazenamento é a confiabilidade das máquinas disponíveis. Desta forma, evita-se a perda de arquivos por falhas na infraestrutura da federação de nuvens.

Atualmente, a política ZooClouS baseia sua medida de confiabilidade com base no *uptime*, porém o faz de maneira direta, de maneira que a confiabilidade cresce ilimitadamente de maneira linear. Como as outras medidas, tais como latência e banda são

limitadas pelo meio físico onde os dados trafegam, a confiabilidade rapidamente se torna o fator principal para as tomadas de decisão da política.

Para evitar a sobreposição do *uptime* diante os demais fatores utilizados no cálculo, foi decidido utilizar uma função logarítimica para diminuir o crescimento da confiabilidade ao longo do tempo. Desta forma, a confiabilidade de um servidor cresce rapidamente nas primeiras horas, reduzindo gradativamente a taxa de crescimento com o tempo. Assim, foi criada a medida *Confiability*, vista na Equação 4.5.

$$Confiability = \ln(Uptime) \tag{4.5}$$

Como a função logarítimica não se inicia da origem ($\ln 0 = -\infty$), foi necessária uma pequena adaptação para que a confiabilidade no tempo t=0 fosse também 0, mostrada na Equação 4.6.

$$Confiability = \ln(Uptime + 1) \tag{4.6}$$

Como pode ser visto na Tabela 4.2, a função logarítimica, apesar de limitar o crescimento da confiabilidade, também torna seus valores extremamente pequenos. Desta forma, foi necessário fazer um ajuste, de modo a tornar os dados comparáveis, conforme mostrada na Equação 4.7

Tabela 4.2: Faixas Usuais de Banda, Latência e *Uptime*

| *Bandwidth* | *Latency* | *Uptime* | $\ln(Uptime + 1)$ |
|---|---|---|---|
| 150 - 200 Mb/s | 0.5 - 2.0 ms | 0 - 168 h | 0 - 2.23 |

$$Confiability = 5 \times \ln(Uptime + 0.4) + 5 \tag{4.7}$$

Desta forma, o *Uptime* influencia mais quando os recursos entram no ar, diminuindo sua influência a medida que cresce a estabilidade dos servidores. Na Figura 4.5 é possível ver o impacto que o tempo (em horas) possui no cálculo da confiabilidade da política BioCirrus ao longo de um dia.

Na Tabela 4.3, é possível analisar melhor o crescimento da confiabilidade, pois após 24h o valor da confiabilidade é de 20.9, e ao aumentar o *Uptime* em 10 vezes, o aumento é de apenas 55%. E esta taxa fica cada vez menor com o passar do tempo, por exemplo de 10 dias para 100 dias, a confiabilidade aumentou apenas 35%, e isso é pouco maior do que o dobro de 1 dia. Passados 1000 dias (2.74 anos), a confiabilidade ainda não chegou ao triplo do que era após 1 dia.

Tabela 4.3: Valores de Confiabilidade

| *Uptime* | 1 dia | 10 dias | 100 dias | 1000 dias |
|---|---|---|---|---|
| Confiabilidade | 20.9 | 32.4 | 43.9 | 55.4 |
| Crescimento relativo | - | 55% | 35% | 26% |

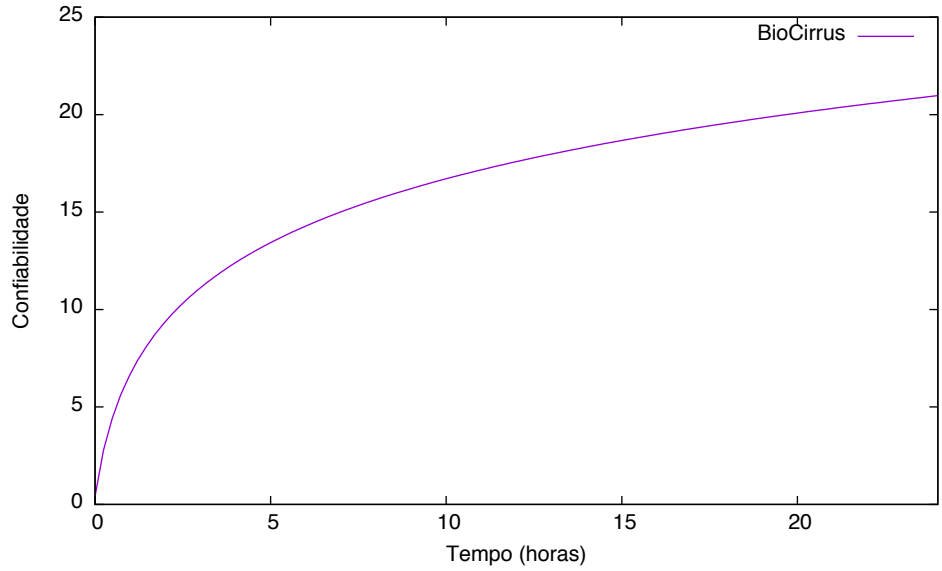


Figura 4.5: Curva de Confiabilidade no Decorrer de 24 Horas.

Com base nestes resultados, bem como os apresentados anteriormente acerca da qualidade do enlace, divisão e compactação de arquivos, foi definida a política BioCirrus, apresentada na Seção 4.3.

## 4.3 Política de Armazenamento BioCirrus

Quando um cliente solicita armazenamento de arquivos na plataforma BioNimbuZ, é calculado um custo para o armazenamento. Este custo ocorre de maneira transparente entre o cliente e os servidores *online* que possuem capacidade para guardar este arquivo. Para a política BioCirrus, cinco variáveis são levadas em consideração para processar este custo, as quais são:

- *Uptime*: Tempo, em horas, em que um servidor se encontra *online* no sistema. É um indicativo de confiabilidade do recurso computacional disponível;
- *Confiability*: Indicativo de confiabilidade baseado no *Uptime*;
- *Latency*: Latência, em milissegundos, para o servidor de destino. Também é importante por medir o atraso do envio;
- *Bandwidth*: Como foi explicado na Seção 4.2.1, a largura de banda demonstrou grande relevância no tempo de transferência de grandes arquivos. Por esta razão, será a variável com maior peso na decisão da política BioCirrus. Ela é medida em Mb/s.
- *Costs*: Valor cobrado por *gigabyte* armazenado.

Para cada servidor é definido o custo de armazenamento, chamado de *storagecost*. Este custo é recalculado a cada envio de arquivo. As quatro variáveis são relacionadas de acordo com a Equação 4.8.

$$S = \left(\frac{L}{B + C}\right) + Costs \tag{4.8}$$

Onde $S$ é o custo total do armazenamento, $L$ é o valor de *Latency*, $B$ é o valor da largura de *Bandwidth* e $C$ é o valor de *Confiability*. Além disso, cada variável possui um peso distinto, com o intuito de ordenar a importância de cada valor. Esses pesos devem obedecer a Equação 4.9.

$$\alpha + \gamma + \beta = 1.0 \tag{4.9}$$

Após diversos testes, estes pesos foram definidos de maneira empírica, sendo $\alpha$ o peso dado à variável *Confiability*, possuindo o valor 0.15; $\gamma$ o peso dado ao custo por Gb, valendo 0.25; e, por último, $\beta$ é dado à largura de banda, valendo 0.6. De acordo com esses pesos, tem-se a Equação 4.10.

$$S = \left(\frac{L}{(\alpha \times C) + (\beta \times B)}\right) + (\gamma \times Costs) \tag{4.10}$$

Expandindo a medida de confiabilidade conforme visto na Seção 4.2.4, verificou-se a Equação 4.11, onde a variável $U$ se refere ao *Uptime*.

$$S = \left(\frac{L}{\alpha \times (5 \times \ln(U + 0.4) + 5) + (\beta \times B)}\right) + (\gamma \times Costs) \tag{4.11}$$

Além da definição do local onde será enviado o arquivo, a política BioCirrus possui uma política de compressão de arquivos, baseada na Equação 4.3, explicada na Seção 4.2.2.

A Figura 4.6 demonstra a relação tempo (eixo y) pela largura de banda (eixo x) e seus pontos de interseção, onde os compactadores usados são representados pelas linhas. De acordo com os resultados, foi montada a Tabela 4.4, onde são mostrados os compactadores testados, assim como o intervalo de faixa de banda aconselhável para seus respectivos usos, mostrando o valor mínimo de banda para uso e o valor de faixa ideal.

Como verificado na Seção 4.2.3, arquivos menores possuem uma compressão menos eficiente, baseado na relação entre o tempo e a taxa de compressão. Portanto, apenas arquivos superiores a 10 MB são compactados, de modo que o arquivo compactado seja maior que 4MB, possuindo uma transferência mais estável, como mostrado na Figura 4.2, o que evita um *overhead* computacional com ganhos muito baixos.

Após a análise dos gráficos e resultados obtidos nos testes, a nova política definiu o compactador mais eficiente a ser utilizado baseado na *bandwidth*, além de avaliar a necessidade de compressão de um arquivo antes de seu envio. Além desses pontos, a política BioCirruS adota o cálculo da largura de banda como ponto primordial para cálculo de armazenamento. Dessa forma, ela comprovou ser uma política bastante eficiente, além de otimizar a capacidade da rede ao compactar os arquivos.

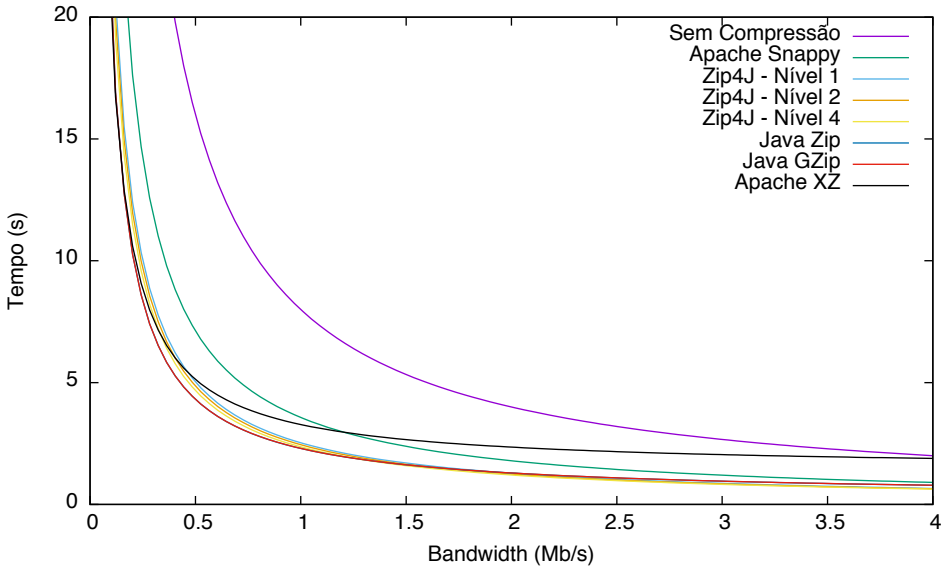


Figura 4.6: Curvas de Tempo para todos os Compactadores.

Tabela 4.4: Faixas de Banda Ótimas para cada Compactador.

| Compactador | Banda Mínima (Kb/s) | Banda Máxima (Kb/s) |
|---|---|---|
| Apache XZ | 0 | 295 |
| Java GZip | 296 | 919 |
| Java Zip | 920 | 3941 |
| Zip 4J - Nível 4 | 3942 | 16174 |
| Zip 4J - Nível 2 | 16174 | 22807 |
| Zip 4J - Nível 1 | 22808 | 102399 |
| Apache Snappy | 102400 | 510976 |
| Sem Compactação | 295957 | $\infty$ |

## 4.4 Considerações Finais

Este capítulo apresentou a política de armazenamento BioCirrus, que tem como objetivo diminuir ao máximo o tamanho e o tempo de transferência dos arquivos, uma vez que arquivos de bioinformática são extremamente grandes. Desta forma, os recursos de rede são economizados e a execução do *workflow* é executada mais rapidamente.

A política BioCirrus foi comparada com as políticas ZooClouS e Ricardo Gallon, e os resultados são apresentados no Capítulo 5.

# Capítulo 5

# Resultados

Neste capítulo são descritos os estudos de performance da política BioCirrus, realizados de forma comparativa às políticas ZooClouS e Ricardo Gallon na plataforma BioNimbuZ. Para isso, foram criadas nuvens ao redor do mundo, onde foram feitos estudos comparativos de tempo de envio dos arquivos, com e sem compactação, bem como tomadas de decisão do local para onde os arquivos deveriam ser enviados.

Como as políticas ZooClous e Ricado Gallon possuem características bastante diferentes, a política BioCirrus foi comparada separadamente com cada uma delas. Em cada comparação, foram feitas duas análises: a diferença no tempo de envio e as decisões tomadas por cada política.

A localização das máquinas pode ser vista na Tabela 5.1. Dessa forma, foram iniciadas duas máquinas na Oceania, sete na Ásia, quatro na Europa, seis na América do Norte e uma na América do Sul. Ao longo dos testes foram adicionadas e removidas diversas máquinas, de modo a verificar a reação de cada política às mudanças de configuração da federação.

Tabela 5.1: Máquinas por Localidade.

| Localização | Número de Máquinas | Nuvem |
|---|---|---|
| Austrália | 2 | Amazon |
| Japão | 4 | Amazon |
| Singapura | 3 | Amazon |
| Alemanha | 2 | Amazon |
| Irlanda | 2 | Amazon |
| Califórnia | 1 | Amazon |
| Óregon | 3 | Amazon |
| Virgínia | 2 | Amazon |
| Brasília | 1 | UnB |

Serão feitas primeiro as comparações com a política ZooClouS e, em seguida, as comparações com a política Ricardo Gallon.

## 5.1 Políticas BioCirrus x ZooClouS

As políticas BioCirrus e ZooClouS possuem em comum as medidas de latência, custo de armazenamento e *uptime*, apesar de tratarem estas medidas de forma distinta. As medidas contrastantes são a largura de banda, que é aferida apenas na política BioCirrus e *free space*, que é utilizada apenas pela política ZooClouS. A seguir, será verificado como ambas as políticas se comportam tanto em relação ao tempo de envio dos arquivos como às decisões para os locais onde deveria ser realizada a transferência.

### 5.1.1 Tempo de Envio

Para se comparar as políticas foram realizados diversos testes, analisando o tempo total de envio dos mesmos arquivos pelas diferentes políticas para a mesma região geográfica. Os arquivos usados nestes testes são arquivos reais de bioinformática, providos por execuções anteriores da plataforma BioNimbuZ.

#### Para os EUA

O primeiro teste se refere ao envio de arquivos para a mesma localidade, focando na melhoria do tempo de envio que a compactação dos arquivos trouxe à política proposta neste trabalho. Para realizar este teste, as máquinas foram iniciadas simultaneamente e com as mesmas configurações de espaço em disco. O teste foi realizado desta forma para minimizar os efeitos que o *uptime* e o *free space* poderiam ter na escolha de ambas as políticas referentes ao destino dos arquivos.

A Figura 5.1 mostra no eixo y o tempo gasto para enviar os arquivos descritos no eixo x. Foram enviados três arquivos reais de bioinformática, e o tempo gasto para a transferência de cada um deles pela política ZooClouS está nas colunas à esquerda. O tempo gasto com a compactação e a transferência usando a política BioCirrus esta à direita. É possível verificar que o tempo gasto pela política BioCirrus é consideravelmente menor, com uma transferência muito mais eficiente. Com estes resultados foi possível verificar que a compressão de arquivos baseado na largura de banda, em situações reais, traz melhorias expressivas para o armazenamento no BioNimbuZ, como foi estudado na Seção 4.2.2.

Além disso, como os fatores *uptime* e *free space* foram desconsiderados, ambas as políticas tomaram decisões muito próximas pois todos os arquivos foram enviados aos EUA em 100% dos casos. Logo, a diferença principal entre as políticas foi a compactação dos arquivos antes do envio, e resultou em uma melhora de aproximadamente 47% no tempo de envio, como pode ser visto na Figura 5.1.

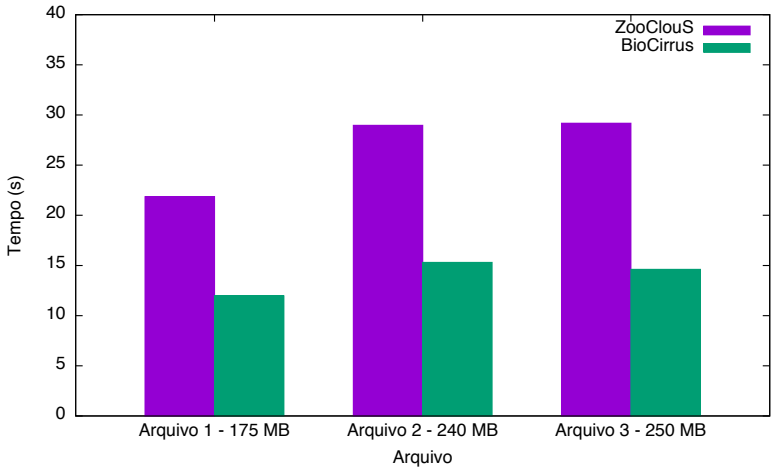


Figura 5.1: Tempo de Transferência para os EUA.

### Para a Europa

Para o segundo teste foram desativadas as máquinas dos EUA, de modo a tornar a transferência dos arquivos mais longa, uma vez que os arquivos deveriam trafegar uma distância física maior. Nesta situação, como mostrado na Figura 5.2, a diferença de eficiência entre as políticas se torna ainda mais evidente. Devido ao fato da taxa de transferência ser menor do que a anterior, o envio de um arquivo compactado se torna ainda mais eficiente, fazendo com que a política BioCirrus gaste em média 53% menos tempo do que a política ZooClous [26].

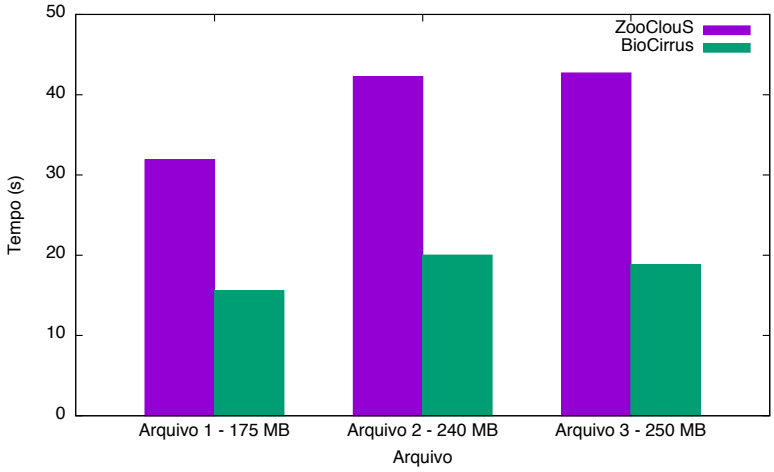


Figura 5.2: Tempo de Transferência para a Europa.

Nesta situação, outro fator que contribuiu para a melhoria do tempo de envio foi que os arquivos foram enviados à países diferentes. Enquanto a política ZooClouS decidiu enviar os arquivos, principalmente, para a Alemanha, a política BioCirrus preferiu enviá-los à Irlanda. Esta decisão aconteceu porque a política BioCirrus leva em conta a largura

de banda, que é maior no enlace para a Irlanda do que no enlace para a Alemanha. Esta decisão será melhor abordada na Seção 5.2.2.

#### Para a Oceania

Após verificar que o tempo de envio pela política BioCirrus é ainda mais eficiente para longas distâncias, foi realizado um último teste focado no tempo de envio, utilizando apenas uma máquina na Austrália, escolhida por possuir a pior latência e largura de banda. Os resultados obtidos com este teste podem ser vistos na Figura 5.3.

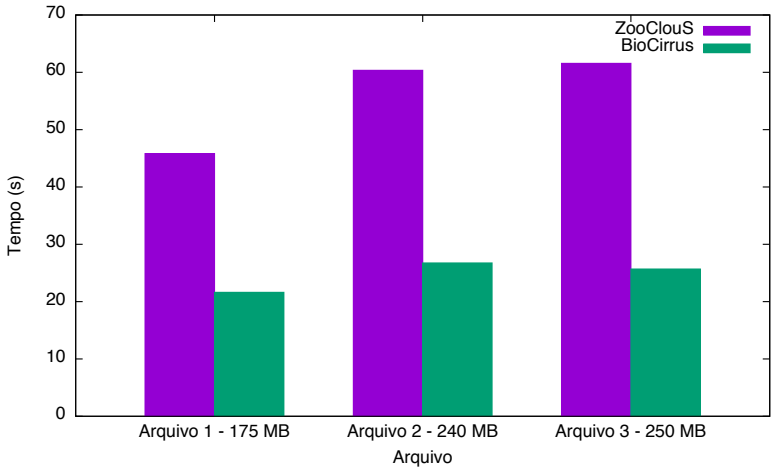


Figura 5.3: Tempo de Transferência para a Oceania.

Como é possível verificar pela Figura 5.3, a melhoria no tempo de envio foi de 56%, em média. Em especial, destaca-se a melhoria no envio do Arquivo 3, o maior entre todos, que possuiu uma melhora de 58%. Este resultado confirma, em uma situação real, o estudo realizado na Seção 4.2.3, de que a compactação para transferência é mais eficiente em arquivos maiores.

Além dos resultados bastante positivos acerca do tempo de transferência dos arquivos compactados, a política BioCirrus se difere da ZooClous nas tomadas de decisão, como foi verificado no segundo teste. Na Seção 5.2.2, será visto com mais detalhes como as duas políticas se comportam acerca das escolhas de servidores para realizar o envio dos arquivos.

### 5.1.2 Tomadas de Decisão

Para analisar como ambas as políticas se utilizam de medidas distintas para tomar decisões acerca do local onde os arquivos serão enviados, foram efetuados testes explorando as medidas únicas entre as políticas. Os testes focaram, então, nas diferenças de *bandwidth*, *free space* e *uptime*.

#### Diferenças de *Bandwidth*

Como abordado na seção anterior, apenas por estimar melhor a qualidade do enlace, a política BioCirrus já toma deciões mais interessantes. No segundo teste, ao transferir

os arquivos para a Europa, a política ZooClouS preferiu enviar os arquivos às máquinas na Alemanha, enquanto a política BioCirrus efetuou a transferência mais vezes para a Irlanda, como pode ser visto na Figura 5.4.

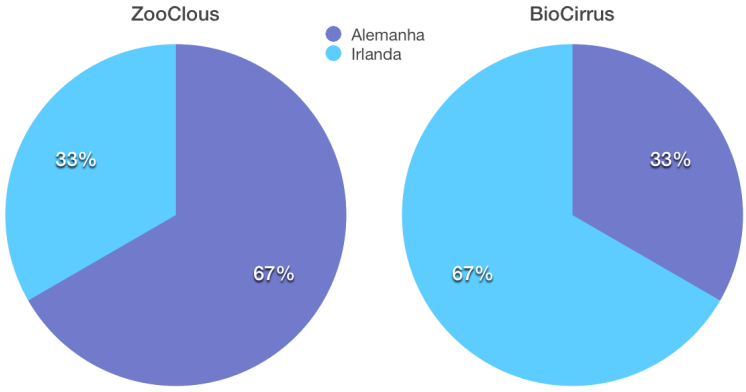


Figura 5.4: Locais de Transferência para a Europa.

A Figura 5.4 mostra à esquerda o percentual de vezes que a política ZooClouS enviou os arquivos para cada localidade. À direita, mostra o mesmo percentual, porém, para a política BioCirrus.

Esta decisão ocorreu porque, apesar da conexão com a Alemanha possuir uma latência média ligeiramente menor, a conexão com a Irlanda possui uma largura de banda maior, como pode ser vito na Tabela 5.2. Assim, como a política BioCirrus é capaz de verificar a qualidade do enlace com mais precisão, é capaz de tomar decisões melhores acerca do local de envio dos arquivos.

Tabela 5.2: Latência e Largura de Banda Médias nos Servidores da Europa.

| Localização | Latência Média | *Bandwidth* Média |
|---|---|---|
| Alemanha | 1.489 ms | 124 Mb/s |
| Irlanda | 1.542 ms | 132 Mb/s |

### Diferenças de *Free Space*

Outro fator que afeta as decisões da política ZooClouS é o espaço livre em disco. A política BioCirrus optou por não utilizar essa medida, uma vez que a lista de servidores é filtrada antes de ser enviada para a política de armazenamento, contendo apenas os servidores capazes de suportar o arquivo.

Para demonstrar como esta medida pode interferir negativamente na escolha do servidor, foi realizado um teste a partir de um servidor na Virgínia. Desta forma, o servidor da UnB tinha 20 GB disponíveis, e o servidor no Japão, 2 TB disponíveis, ambos os servidores foram iniciados aproximadamente ao mesmo tempo, de modo a diminuir a interferência do *uptime* no cálculo de ambas as políticas.

Desta forma, apesar da latência ser superior, a política ZooClouS optou por enviar o segundo arquivo dos EUA para o Japão, enquanto a política BioCirrus os enviou para

Brasília, na nuvem da UnB. Como o arquivo ocupava pouco mais de 240 MB, ambas as máquinas possuiam espaço suficiente para armazená-los. A decisão da política ZooClouS, como pode ser vista na Figura 5.5, com destaque para o Arquivo 2, causou um enorme aumento no tempo de transferência.

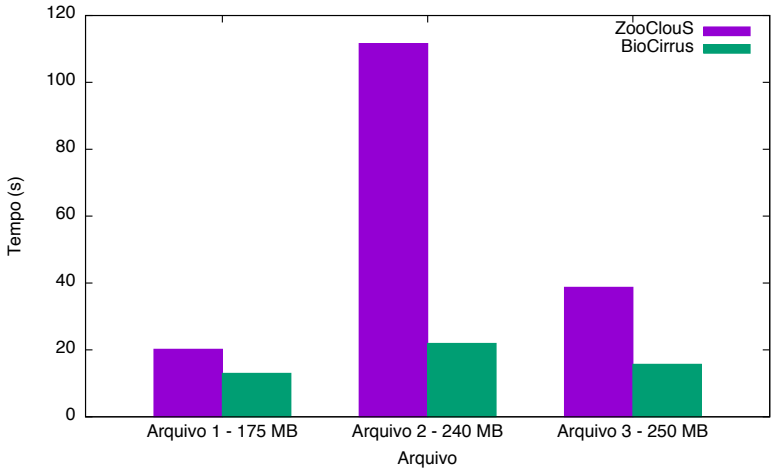


Figura 5.5: Interferência do *Free Space* no Tempo de Transferência.

Ao escolher enviar o arquivo dos EUA para o Japão, ao invés do Brasil, a política ZooClouS gastou 500% do tempo gasto pela política BioCirrus. E esta decisão ocorreu sem necessidade, uma vez que ambas as máquinas possuiam espaço suficiente para armazenar o arquivo. Além do *free space*, a política BioCirrus também utiliza a variável *uptime* de maneira distinta à ZooClouS, como será visto a seguir.

### Diferenças de *Uptime*

A política ZooClouS, assim como a BioCirrus, utiliza o *uptime* como medida de confiabilidade de uma máquina. No entanto, a política ZooClouS permite que essa medida cresça indefinidamente de maneira linear, considerando que uma máquina ativa há um mês é duas vezes mais confiável que uma máquina ativa há quinze dias.

Para demonstrar que o crescimento linear ilimitado do *uptime* é prejudicial, foi realizado um teste, no qual a transferência se iniciou na Califórnia, com servidores disponíveis na Singapura, ativos há uma semana (180 h); e na Irlanda, ativos há 60 horas. Todos os servidores possuiam o mesmo *free space*, de forma a evitar a interferência do mesmo nos testes.

De acordo com a política ZooClouS, os servidores da Irlanda são três vezes mais confiáveis, enquanto a política BioCirrus considera que, apesar da confiabilidade do servidor na Singapura ser maior do que a do servidor na Irlanda, isso não é uma relação direta, e a diferença de confiabilidade é de apenas 17%.

Desta forma, utilizar o *uptime* de maneira linear no cálculo interfere bastante na decisão, e faz com que os arquivos sejam enviados pela política ZooClouS à Singapura, enquanto a política BioCirrus considerou que o servidor na Irlanda era confiável o suficiente, e mandou os arquivos para lá. Esta decisão, obviamente, impactou no tempo de envio dos arquivos, como mostrado na Figura 5.6.

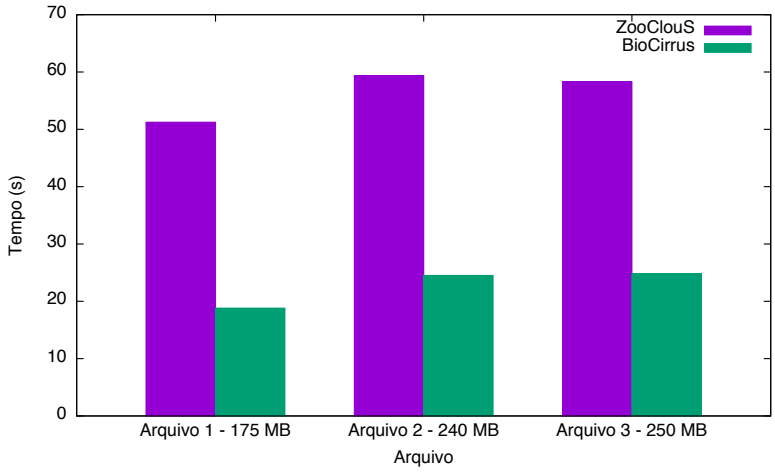


Figura 5.6: Interferência do *Uptime* no Tempo de Transferência.

Assim sendo, nesse teste a política BioCirrus diminuiu o tempo de envio da política ZooClouS, em média, 60%. Comparando com o primeiro teste, em que a diferença entre as duas políticas foi de 47%, nota-se que o *uptime* interfere muito mais na política ZooClouS do que na BioCirrus.

Como pode ser notado, os resultados apresentados nesta seção mostraram que a política BioCirrus alcança desempenhos muito superiores à política ZooClouS relativos ao tempo de transferência e à localidade dos arquivos. Diversos fatores contribuíram para esta melhoria, os principais foram:

- Acrescentar a compressão dos arquivos antes do envio;
- Melhorar a estimativa de qualidade do enlace por meio da largura de banda;
- Eliminar o impacto do *free space* em suas decisões;
- Diminuir o impacto do *uptime* quando os servidores já estão estáveis.

Na próxima seção, serão mostrados os resultados da comparação entre as políticas BioCirrus e Ricardo Gallon.

## 5.2 Políticas BioCirrus x Ricardo Gallon

As políticas BioCirrus e Ricardo Gallon possuem apenas a latência e o custo de armazenamento como medida em comum para a escolha do local de armazenamento. A política BioCirrus se utiliza também da largura de banda e *uptime*, enquanto a política Ricardo Gallon se utiliza da carga de trabalho e do número de processadores. Nas próximas seções, será analisado o desempenho das políticas em relação ao tempo de envio e às decisões do local de armazenamento do arquivo.

### 5.2.1 Tempo de Envio

Para analisar como ambas as políticas se comportam, foram realizados diversos testes com o envio de arquivos reais de bioinformática. Estes arquivos foram transferidos à partir da nuvem da UnB, em Brasília, para a nuvem da Amazon em Óregon, nos EUA. As transferências foram realizadas alternando as políticas, sem intervalo entre os envios.

#### Sem Limite de Conexão

Para realizar o primeiro teste, ambas as máquinas operaram sem restrição alguma sobre a largura de banda ou latência. Desta forma, foi possível verificar o comportamento das políticas em uma situação real.

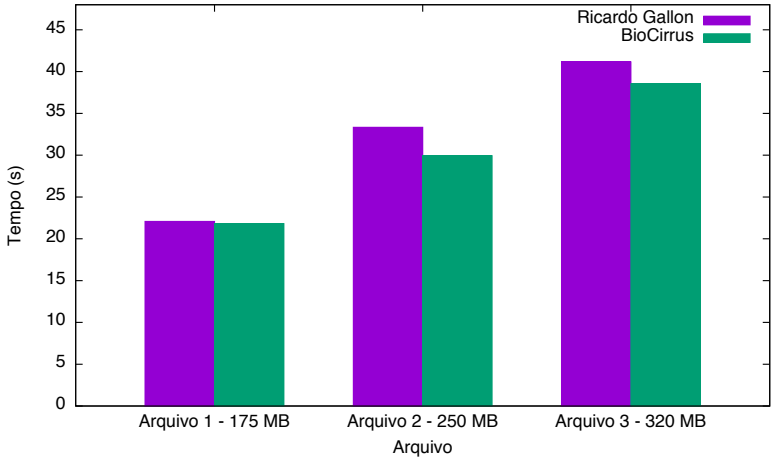


Figura 5.7: Tempo de Transferência sem Limites de Conexão.

Assim sendo, como é possível visualizar na Figura 5.7, ambas as políticas obtiveram um tempo próximo, com os tempos da política BioCirrus sendo 6% menores na média. Esta diferença ocorre pelo fato de que, apesar de ambas utilizarem a compactação de arquivos, a política BioCirrus calcula a largura de banda e opta pela estratégia de compressão mais adequada para a situação, enquanto a política Ricardo Gallon sempre utiliza o mesmo compactador.

#### Com Limites de Conectividade

Em situações de envio de arquivos entre diferentes nuvens, não é raro que haja um limite na velocidade de conexão, ou mesmo que transferências em altas velocidades sejam cobradas. Desta forma, foram feitos testes comparando ambas as políticas na mesma situação que o primeiro teste, com o adicional de um limite de *upload* do recurso da UnB de 100 Mbps.

Desta forma, na Figura 5.8 foi possível verificar que a diferença média dos tempos de envio entre as políticas aumentou, sendo a política BioCirrus, em média 27% mais rápida do que a política Ricardo Gallon. Este resultado corrobora o anterior, e mostra que a estratégia de compactação baseada na largura de banda possui resultados mais interessantes que a estratégia fixa.

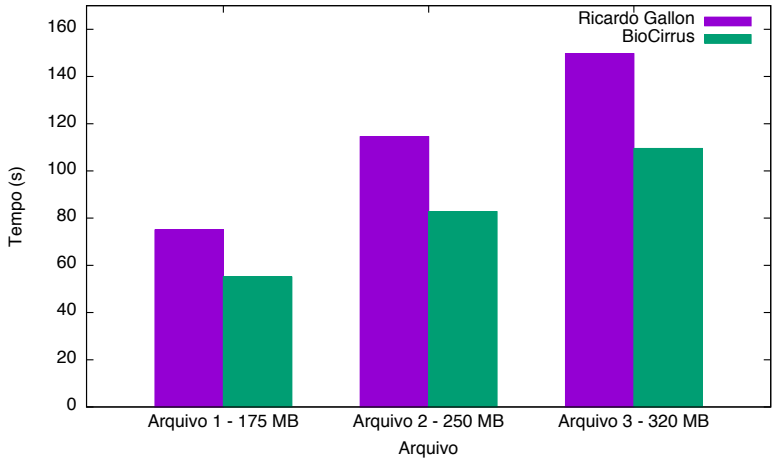


Figura 5.8: Tempo de Transferência com Limite de 100 Mbps.

Além disso, reduzindo-se ainda mais a velocidade para 50, 25, 12.5 e 6.25 Mbps, é possível observar que a situação apresentada se mantêm. Esta situação é ilustrada na Figura 5.9, na qual mostra-se que a escolha do compactador feita pela política BioCirrus garante que, quanto menor for a velocidade de conexão, maior será a diferença percentual do tempo gasto por ambas as políticas.

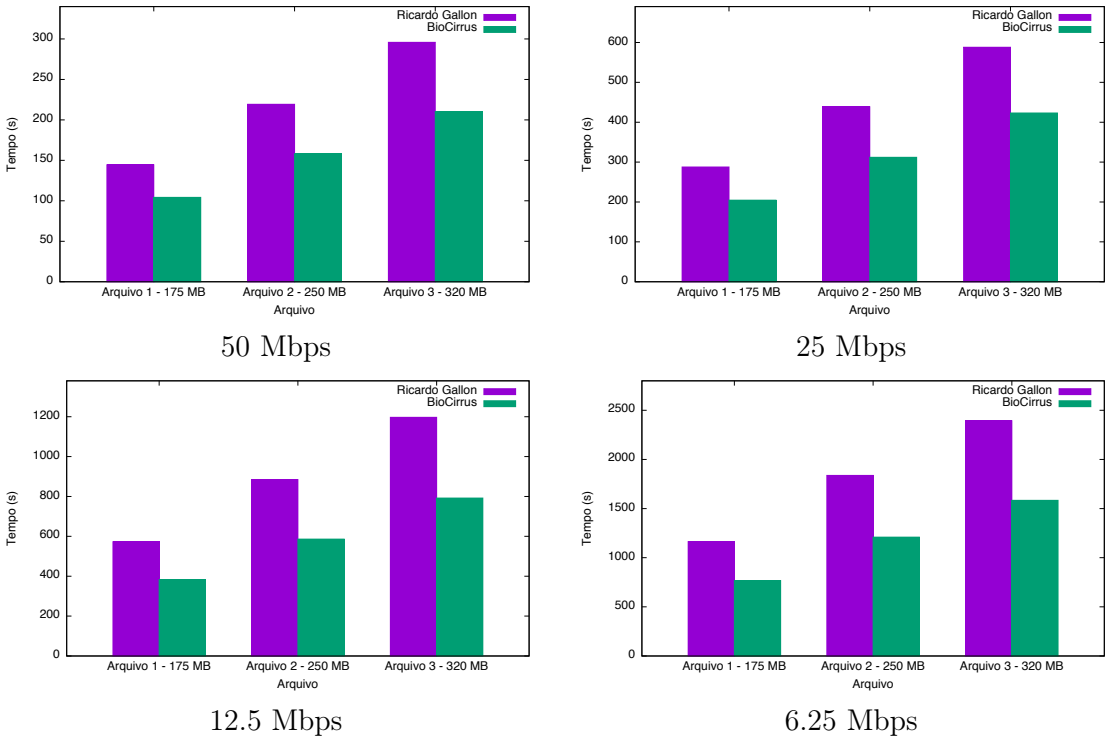


Figura 5.9: Tempo de Transferência para Diferentes Limites de Conexão.

Na próxima seção será analisada a maneira como as diferentes métricas utilizadas por ambas as políticas afetam as decisões do local de armazenamento.

### 5.2.2 Tomadas de Decisão

Ao analisar o desempenho das políticas é crucial verificar as decisões que foram tomadas com base nas medidas utilizadas, de modo a aumentar ou diminuir a influência dos fatores discrepantes no cálculo. Desta forma, foram analisados os fatores número de processadores, utilizado pela política Ricardo Gallon, e *uptime*, utilizado pela política BioCirrus.

#### Número de Processadores

A política Ricardo Gallon utiliza a medida de quantidade de núcleos para estimar onde a política de escalonamento irá utilizar o arquivo. Como foi explanado na Seção 3.3.1, esta estimativa foge do escopo de uma política de armazenamento, e possui efeitos colaterais indesejáveis. Desta forma, foram feitos testes com o arquivo sendo transferido da Virgínia, nos EUA, para duas possíveis máquinas em Frankfurt, na Alemanha. Uma das máquinas possuia somente um núcleo, com custo por hora de U$0.50, enquanto a outra possuía dois núcleos, com custo de U$2.0 por hora.

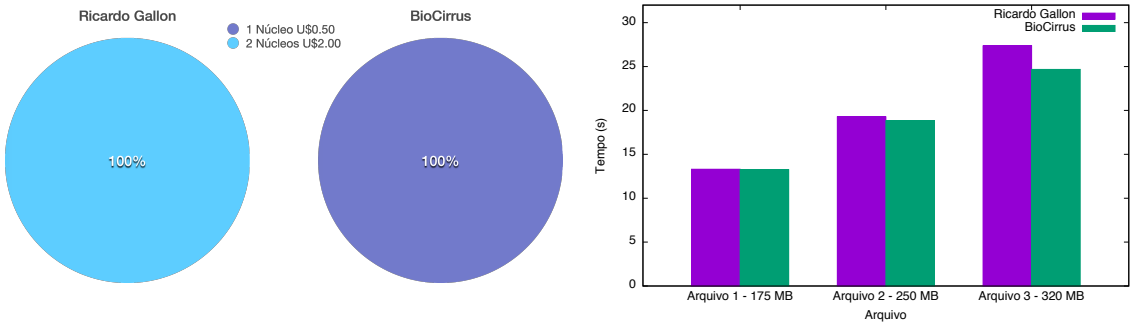


Figura 5.10: Comparativo de Transferência com Diferenças de Hardware.

Como pode ser visto no gráfico à esquerda da Figura 5.10, a política Ricardo Gallon optou por enviar o arquivo para a máquina mais cara, mesmo que não houvesse melhoria alguma no tempo de transferência, como pode ser visto no gráfico à direita da mesma imagem. Esta tentativa de prever o comportamento de outra política é bastante desvantajosa, pois não se sabe se, efetivamente, se a política de escalonamento vai realizar o processamento no local escolhido para o armazenamento.

#### *Uptime*

Outra medida distinta entre as políticas é o *uptime*, que é usado pela política BioCirrus para estimar a confiabilidade de um recurso computacional, diminuindo assim a chance de enviar os dados para uma máquina instável e perder os dados obtidos pela execução.

Para verificar como o *uptime* afeta ambas as políticas, foram feitos testes com o envio de arquivos de uma máquina em Óregon, nos EUA, para duas possíveis máquinas,

idênticas em configuração. A primeira estava localizada em Sydney - Austrália, com um *uptime* de 180 horas, enquanto a outra tinha acabado se ser iniciada em Singapura.

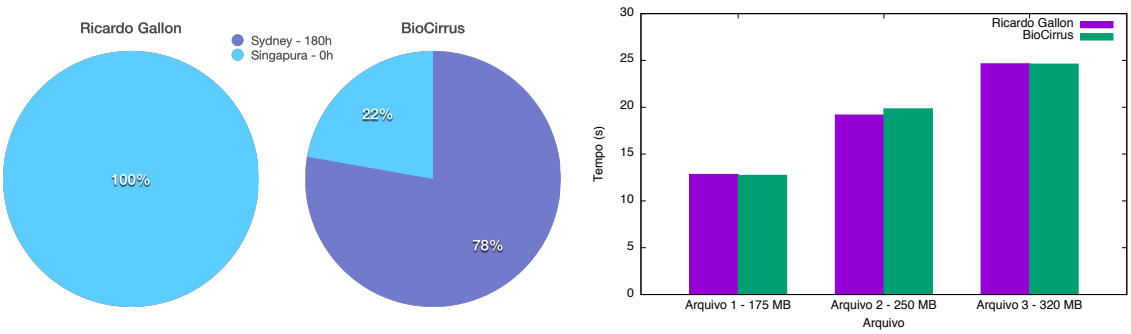


Figura 5.11: Comparativo de Transferência com Diferenças de *Uptime*.

Como pode ser visto no gráfico à esquerda da Figura 5.4, a política Ricardo Gallon preferiu enviar seus dados à Singapura, uma vez que a latência média para aquela localidade é menor do que a latência para Sydney. Porém, a confiabilidade de uma máquina que acabou se ser iniciada deve ser muito inferior à de uma máquina ativa há mais de uma semana. Apesar da política Ricardo Gallon ter feito sua escolha baseada, principalmente, na rede disponível, é possível ver no gráfico à direita da Figura 5.4 que houve pouca mudança no que diz respeito ao tempo de transferência de ambas as políticas.

Como apresentado nessa seção, os resultados demonstram que a política BioCirrus alcança melhores desempenhos que a política Ricardo Gallon no que diz respeito à transferência de arquivos compactados, principalmente, quando há limitações na velocidade máxima de transferência. Além desses resultados, mostrou-se que algumas decisões da política BioCirrus são mais interessantes ao se analisar o preço e a confiabilidade dos recursos disponíveis. Os fatores que contribuíram para esta melhoria foram:

- Manter a compressão dos arquivos antes do envio, mas otimizá-la de acordo com a rede;
- Acrescentar a estimativa de qualidade do enlace por meio da largura de banda;
- Acrescentar um fator de confiabilidade, baseado no *uptime*;
- Eliminar o impacto que medidas de escalonamento possuíam sobre a política.

## 5.3 Considerações Finais

Como visto nas Seções 5.1 e 5.2, a política BioCirrus possui desempenho superior às políticas já propostas para a plataforma BioNimbuZ, ZooClouS [26] e Ricardo Gallon [17]. Além disso, a BioCirrus considera apenas os fatores essenciais para uma boa política de armazenamento, o que faz com que suas decisões, acerca do local de armazenamento, sejam bastante interessantes quando comparadas com as políticas anterioes.

Desta forma, foi alcançado o objetivo principal deste trabalho, que foi propor uma nova política para a plataforma BioNimbuZ mais eficiente que as anteriores. No Capítulo 6 será feito um apanhado geral deste trabalho, mostrando suas aplicações e trabalhos futuros.

# Capítulo 6

# Conclusão

Neste trabalho foi proposta uma nova política de armazenamento para a plataforma BioNimbuZ, chamada BioCirrus. O algoritmo foi baseado em diversas hipóteses, apresentadas no Capítulo 4, e nas atuais políticas de armazenamento do BioNimbuZ, descritas na Seção 3.3.

Definir os pontos em que a nova política é baseada foi essencial para a melhoria proposta. Para a plataforma BioNimbuZ, a qualidade do uso da rede e da taxa de transferência é de suma importância, visto que no BioNimbuZ são armazenados arquivos grandes. Para que isto acontecesse, foi preciso aferir as taxas de transferência entre as máquinas disponíveis na federação e a máquina de envio do arquivo. Além de prezar pela qualidade de rede da federação, a nova política propôs a compactação de arquivos antes de seu envio, um estudo para validar a hipótese de quebra dos arquivos e uma maneira eficiente de se estimar a confiabilidade de um recurso computacional ao longo do tempo.

A política BioCirrus trouxe melhorias no tempo de envio, que em alguns testes chegou a ser reduzido pela metade, e nas escolhas dos locais de armazenamento. Além disso, ela demonstrou ter a maior eficiência de um algoritmo baseado em grande parte no valor da taxa de transferência, e que compacta os arquivos a serem enviados. Assim, outros fatores, como latência e espaço em disco disponível, perderam peso na escolha do servidor para o armazenamento.

Para trabalhos futuros, alguns pontos ainda devem ser estudados. Um deles é reestipular o valor de custo do armazenamento representado na fórmula pela variável *costs*. Atualmente, este valor é o custo por gigabyte, e não o custo efetivo do armazenamento daquele arquivo. Propõem-se que este valor seja alterado pelo custo do arquivo que será armazenado na nuvem. Outra proposta futura é a adição de ferramentas de compressão otimizadas para arquivos de bioinformática, de modo a tornar o tempo de envio ainda menor.

Outro ponto importante é o estudo dos protolocos de transferência, para que seja viável uma implementação de uma Interface de Gerenciamento de Dados em Nuvem (CDMI - *Cloud Data Manage Interface*). A transferência atual funciona por meio de *scp (Security Copy)*. A interface CDMI trabalha com vários protocolos de transferência e seleciona o melhor método disponível na arquitetura, para efetuar a transferência. Assim, é sugerido um modelo mais dinâmico, com capacidade de otimizar a transferência baseando-se em alguns fatores da rede.

# Referências

[1] Emerson Alecrim. Entendendo a cloud computing. `http://www.infowester.com/cloudcomputing.php`, 2008. 7

[2] Amazon. Amazon elasticache. `https://aws.amazon.com/pt/elasticache/`, 2006. 9

[3] Amazon. Amazon s3. `http://aws.amazon.com/pt/s3/`, 2006. 9

[4] Amazon. Amazon simpledb. `https://aws.amazon.com/pt/simpledb/`, 2007. 9

[5] Amazon. Amazon ebs. `https://aws.amazon.com/pt/ebs/`, 2008. 9

[6] Amazon. Amazon rds. `https://aws.amazon.com/pt/rds/`, 2009. 9

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. `https://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`, 2009. 3

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the $5^{th}$ utility. *Future Generation computer systems*, 2009. 3, 4, 8

[9] Cavas. Modelos de implementação cloud computing. `http://cavas.com.br/gestao/modelos-de-implementacao-cloud-computing/`, 2008. 7

[10] Cavas. Serviços explorados no cloud computing. `http://cavas.com.br/gestao/servicos-explorados-no-cloud-computing/`, 2008. 7

[11] Antonio Celesti, Francesco Tusa, Massimo Villari, and Antonio Puliafito. How to enhance cloud architectures to enable cross-federation. in cloud computing (cloud). *2010 IEEE 3rd International Conference on*, pages 337–345, 2010. 1

[12] Microsoft Corporation. Windows azure. `http://www.windowsazure.com/pt-br/pricing/free-trial/`, 2012. 19

[13] H. H. P. M. Costa. Controle de acesso na plataforma de nuvem federada bionimbuz, 2015. vii, 15

[14] Inc Eucalyptus Systems. Eucalyptus. `www.eucalyptus.com`, 2008. 6

[15] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop*, pages 1–10, November 2008. 4

[16] The Apache Software Foundation. Apache zookeeper. `https://zookeeper.apache.org/`. [Online; accessed 20-July-2015]. 13, 14

[17] Ricardo Fernandes Gallon. Política de armazenamento de dados em nuvens federadas para dados biológicos, 2014. 2, 17, 20, 22, 42

[18] Google. Google app engines. `https://cloud.google.com/appengine/docs`, 2005. 6

[19] Google. Google docs. `https://www.google.com/docs/about/`, 2005. 6

[20] Billy Hoffman. Bandwidth, latency, and the "size of your pipe", December 2011. vii, 22

[21] Intel. Virtualization and cloud computing. `http://www.intel.com/content/dam/www/public/us/en/documents/guides/cloud-computing-virtualization-building-private-iaas-guide.pdf`, 2013. 4

[22] M. T. Jones. Cloud computing platforms and applications. `http://aws.amazon.com/pt/ec2/`, 2008. 3

[23] Amazon Web Services LLC. Amazon elastic compute cloud (ec2). `http://aws.amazon.com/pt/ec2/`, 2012. 4, 6, 19

[24] Manjrasoft Pty Ltd. Aneka. `http://www.manjrasoft.com/aneka_architecture.html`, 2006. 6

[25] Microsoft. Microsoft office 365. `https://products.office.com/pt-br/office-365-home`, 2011. 6

[26] B. R. Moura and D. L. Bacelar. Política para armazenamento de arquivos no zoonimbus, 2013. vii, 12, 13, 17, 18, 19, 20, 22, 34, 42

[27] University of Vienna. Simap - the similarity matrix of proteins. `http://cube.univie.ac.at/resources/simap`. [Online; accessed 04-August-2015]. 11

[28] G. S. S. Oliveira. Acosched - um escalonador para o ambiente de federada zoonimbus, 2013. 12

[29] C. Dovrolis R. Prasad, M. Murray. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network,Volume 17*, pages 27–35, November 2003. 22

[30] Bhaskar Prasad Rimal and Ian Lumb. A taxonomy and survey of cloud computing systems. *Fifth International Joint Conference on INC, IMS and IDC.*, pages 44–51, August 2009. 5

[31] A. Ruiz-Alvarez and M. Humphrey. A model and decision procedure for data storage in cloud computing. *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 572–579, may 2012. 9

[32] H. V. Saldanha. Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para execução de workflows em bioinformática. Master's thesis, Departamento de Ciência da Computação, Universidade de Brasília, 2012. 1, 3, 11, 12, 15

[33] David Salomon. *Data Compression : The Complete Reference*. Springer, Nova Iorque, 2nd edition, 2000. 24

[34] Joseph D. Sloan. *High Performance Linux Clusters: With OSCAR, Rocks, openMosix, and MPI (Nutshell Handbooks)*. O'Reilly Media, 2004. 3

[35] SNIA. Cloud data management interface. 2012. 10

[36] Flávio RC Sousa, Leonardo O Moreira, and Javam C Machado. *III Escola Regional de Computação Ceará, Maranhão e Piauí ERCEMAPI*, dezember. 4, 7

[37] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Paris, 2002. 21

[38] Andrew S. Tanenbaum. *Organização estruturada de computadores*. Prentice Hall, São Paulo, 5th edition, 2007. 3

[39] TechTarget. Storage as a service (saas) definition. `http://searchstorage.techtarget.com/definition/Storage-as-a-Service-SaaS`, 2006. 11

[40] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. *A break in the clouds:Towards a cloud defnition.* SIGCOM Comput. Commum, São Paulo, 5th edition, dezember 2008. vii, 6

[41] Christian Vecchiola, Xingchen Chu, and Rajkumar Buyya. Aneka: A software platform for .net-based cloud computing. *CoRR*, November. vii, 5

[42] E. White, L. McMillan, P. Romanski, M. O'Gara, and J. Bloomberg. Inter-cloud peering points. *CloudExpo: Blog Feed Post*, December 2010. vii, 8, 9

[43] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu. Cloud storage as the infrastructure of cloud computing. *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*, pages 380–383, June 2010. vii, 9, 10

# Anexo I

# Análise de Compressão

Tabela I.1: Resultados da Análise de Compressão.

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| Snappy | - | 499.028 | 128 | 1.128 | 55.009 |
| Snappy | - | 404.790 | 64 | 0.693 | 54.819 |
| Snappy | - | 358.486 | 32 | 0.391 | 54.753 |
| Snappy | - | 351.151 | 16 | 0.201 | 55.256 |
| Snappy | - | 338.519 | 8 | 0.104 | 55.431 |
| Snappy | - | 329.391 | 4 | 0.053 | 55.531 |
| Snappy | - | 313.546 | 1 | 0.014 | 55.556 |
| Snappy | - | 312.511 | 2 | 0.028 | 55.554 |
| Zip4J | 1 | 289.021 | 128 | 2.434 | 68.708 |
| Zip4J | 2 | 234.069 | 128 | 3.048 | 69.689 |
| Zip4J | 1 | 229.827 | 64 | 1.524 | 68.417 |
| Zip4J | 1 | 199.624 | 32 | 0.875 | 68.262 |
| Zip4J | 2 | 186.954 | 64 | 1.902 | 69.467 |
| Zip4J | 1 | 186.102 | 8 | 0.236 | 68.690 |
| Zip4J | 1 | 185.637 | 16 | 0.472 | 68.583 |
| Zip4J | 1 | 182.792 | 4 | 0.120 | 68.748 |
| Zip4J | 4 | 180.218 | 128 | 4.067 | 71.583 |
| Zip4J | 1 | 179.515 | 2 | 0.061 | 68.759 |
| Zip4J | 1 | 173.577 | 1 | 0.031 | 68.746 |
| Zip4J | 2 | 162.675 | 32 | 1.091 | 69.358 |
| Zip4J | 2 | 153.197 | 16 | 0.582 | 69.691 |
| Zip4J | 2 | 152.118 | 8 | 0.293 | 69.792 |
| Zip4J | 2 | 149.319 | 4 | 0.149 | 69.854 |
| Zip4J | 2 | 146.149 | 2 | 0.076 | 69.869 |
| Zip4J | 4 | 142.546 | 64 | 2.554 | 71.115 |
| Zip4J | 2 | 140.936 | 1 | 0.039 | 69.855 |
| Zip4J | 3 | 138.194 | 128 | 5.239 | 70.706 |
| Zip4J | 4 | 123.739 | 32 | 1.465 | 70.830 |
| Zip4J | 4 | 115.520 | 8 | 0.393 | 71.085 |
| Zip4J | 4 | 113.302 | 4 | 0.200 | 71.112 |

Tabela I.1: (Continuação)

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| Zip4J | 4 | 112.409 | 2 | 0.101 | 71.123 |
| Zip4J | 4 | 110.170 | 16 | 0.825 | 71.036 |
| Zip4J | 3 | 109.181 | 64 | 3.308 | 70.556 |
| Zip4J | 4 | 108.721 | 1 | 0.052 | 71.105 |
| Zip4J | 3 | 96.159 | 32 | 1.876 | 70.498 |
| Zip4J | 3 | 90.696 | 16 | 0.999 | 70.847 |
| Zip4J | 3 | 89.982 | 8 | 0.504 | 70.972 |
| Zip4J | 3 | 88.095 | 4 | 0.258 | 71.040 |
| Zip4J | 3 | 87.306 | 2 | 0.130 | 71.049 |
| Zip4J | 3 | 85.257 | 1 | 0.066 | 71.028 |
| Zip4J | 5 | 82.321 | 128 | 8.995 | 72.318 |
| Zip4J | 5 | 65.064 | 64 | 5.653 | 71.837 |
| Zip4J | 5 | 56.401 | 32 | 3.247 | 71.542 |
| Zip4J | 5 | 54.447 | 16 | 1.686 | 71.729 |
| Zip4J | 5 | 52.566 | 8 | 0.873 | 71.769 |
| Zip4J | 5 | 51.557 | 4 | 0.445 | 71.793 |
| Zip4J | 5 | 51.288 | 2 | 0.223 | 71.803 |
| Zip4J | 5 | 50.415 | 1 | 0.113 | 71.785 |
| Apache Deflate | - | 33.579 | 128 | 22.319 | 73.191 |
| Apache GZip | - | 33.414 | 128 | 22.429 | 73.191 |
| Java Zip | - | 32.973 | 128 | 22.778 | 73.349 |
| Java GZip | - | 32.915 | 128 | 22.818 | 73.349 |
| ZLib - Default | 1 | 32.462 | 128 | 23.087 | 73.191 |
| ZLib - Filtered | 5 | 32.423 | 128 | 23.115 | 73.191 |
| ZLib - Default | 2 | 32.389 | 128 | 23.139 | 73.191 |
| ZLib - Filtered | 4 | 32.259 | 128 | 23.232 | 73.191 |
| ZLib - Default | 3 | 32.237 | 128 | 23.248 | 73.191 |
| ZLib - Default | 4 | 32.216 | 128 | 23.263 | 73.191 |
| ZLib - Default | 5 | 32.206 | 128 | 23.271 | 73.191 |
| ZLib - Filtered | 2 | 32.173 | 128 | 23.295 | 73.191 |
| ZLib - Filtered | 3 | 32.151 | 128 | 23.310 | 73.191 |
| ZLib - Filtered | 1 | 32.106 | 128 | 23.343 | 73.191 |
| ZLib - Huffman | 5 | 31.897 | 128 | 23.496 | 73.191 |
| ZLib - Huffman | 4 | 31.688 | 128 | 23.651 | 73.191 |
| ZLib - Huffman | 1 | 31.625 | 128 | 23.698 | 73.191 |
| ZLib - Huffman | 3 | 30.263 | 128 | 24.765 | 73.191 |
| ZLib - Huffman | 2 | 28.651 | 128 | 26.158 | 73.191 |
| Apache Deflate | - | 26.689 | 64 | 13.961 | 72.781 |
| Apache GZip | - | 26.565 | 64 | 14.027 | 72.781 |
| Java Zip | - | 26.161 | 64 | 14.266 | 72.898 |
| Java GZip | - | 26.034 | 64 | 14.336 | 72.898 |
| ZLib - Huffman | 3 | 25.972 | 64 | 14.347 | 72.781 |
| ZLib - Huffman | 2 | 25.950 | 64 | 14.359 | 72.781 |

Tabela I.1: (Continuação)

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| ZLib - Huffman | 5 | 25.944 | 64 | 14.363 | 72.781 |
| ZLib - Huffman | 4 | 25.938 | 64 | 14.366 | 72.781 |
| ZLib - Default | 5 | 25.636 | 64 | 14.535 | 72.781 |
| ZLib - Huffman | 1 | 25.630 | 64 | 14.538 | 72.781 |
| ZLib - Default | 3 | 25.599 | 64 | 14.556 | 72.781 |
| ZLib - Default | 4 | 25.590 | 64 | 14.561 | 72.781 |
| ZLib - Default | 1 | 25.582 | 64 | 14.566 | 72.781 |
| ZLib - Default | 2 | 25.573 | 64 | 14.571 | 72.781 |
| ZLib - Filtered | 5 | 25.497 | 64 | 14.614 | 72.781 |
| ZLib - Filtered | 4 | 25.450 | 64 | 14.641 | 72.781 |
| ZLib - Filtered | 1 | 25.103 | 64 | 14.844 | 72.781 |
| ZLib - Filtered | 3 | 24.996 | 64 | 14.907 | 72.781 |
| Apache Deflate | - | 23.201 | 32 | 8.003 | 72.536 |
| Apache GZip | - | 23.090 | 32 | 8.042 | 72.536 |
| Java Zip | - | 22.822 | 32 | 8.147 | 72.630 |
| Java GZip | - | 22.753 | 32 | 8.171 | 72.630 |
| ZLib - Filtered | 4 | 22.653 | 32 | 8.197 | 72.536 |
| ZLib - Filtered | 5 | 22.652 | 32 | 8.197 | 72.536 |
| ZLib - Huffman | 1 | 22.647 | 32 | 8.199 | 72.536 |
| ZLib - Filtered | 2 | 22.646 | 64 | 16.454 | 72.781 |
| ZLib - Filtered | 3 | 22.642 | 32 | 8.201 | 72.536 |
| ZLib - Huffman | 4 | 22.638 | 32 | 8.202 | 72.536 |
| ZLib - Huffman | 2 | 22.629 | 32 | 8.205 | 72.536 |
| ZLib - Huffman | 5 | 22.629 | 32 | 8.205 | 72.536 |
| ZLib - Filtered | 2 | 22.616 | 32 | 8.210 | 72.536 |
| ZLib - Huffman | 3 | 22.614 | 32 | 8.211 | 72.536 |
| ZLib - Filtered | 1 | 22.592 | 32 | 8.219 | 72.536 |
| Apache Deflate | - | 22.447 | 16 | 4.148 | 72.750 |
| ZLib - Default | 1 | 22.407 | 32 | 8.287 | 72.536 |
| ZLib - Default | 5 | 22.317 | 32 | 8.320 | 72.536 |
| Apache GZip | - | 22.294 | 16 | 4.176 | 72.750 |
| ZLib - Default | 3 | 22.262 | 32 | 8.341 | 72.536 |
| ZLib - Default | 4 | 22.244 | 32 | 8.347 | 72.536 |
| ZLib - Huffman | 4 | 21.923 | 16 | 4.247 | 72.750 |
| ZLib - Huffman | 3 | 21.896 | 16 | 4.252 | 72.750 |
| ZLib - Huffman | 2 | 21.868 | 16 | 4.258 | 72.750 |
| ZLib - Huffman | 1 | 21.859 | 16 | 4.260 | 72.750 |
| ZLib - Filtered | 2 | 21.852 | 16 | 4.261 | 72.750 |
| ZLib - Filtered | 4 | 21.851 | 16 | 4.261 | 72.750 |
| ZLib - Default | 4 | 21.850 | 16 | 4.261 | 72.750 |
| ZLib - Filtered | 5 | 21.847 | 16 | 4.262 | 72.750 |
| ZLib - Huffman | 5 | 21.843 | 16 | 4.263 | 72.750 |
| ZLib - Filtered | 3 | 21.840 | 16 | 4.263 | 72.750 |

Tabela I.1: (Continuação)

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| ZLib - Default | 2 | 21.837 | 16 | 4.264 | 72.750 |
| ZLib - Filtered | 1 | 21.834 | 16 | 4.264 | 72.750 |
| ZLib - Default | 5 | 21.830 | 16 | 4.265 | 72.750 |
| ZLib - Default | 3 | 21.826 | 16 | 4.266 | 72.750 |
| ZLib - Default | 1 | 21.821 | 16 | 4.267 | 72.750 |
| ZLib - Default | 2 | 21.755 | 32 | 8.535 | 72.536 |
| Java Zip | - | 21.656 | 16 | 4.314 | 72.991 |
| Java GZip | - | 21.536 | 16 | 4.338 | 72.992 |
| Apache Deflate | - | 21.528 | 8 | 2.164 | 72.807 |
| Apache GZip | - | 21.462 | 8 | 2.171 | 72.807 |
| Java Zip | - | 21.235 | 1 | 0.281 | 74.787 |
| Java Zip | - | 21.199 | 8 | 2.207 | 73.126 |
| Apache Deflate | - | 21.170 | 4 | 1.101 | 72.839 |
| Java GZip | - | 21.150 | 8 | 2.212 | 73.127 |
| Java Zip | - | 21.142 | 2 | 0.560 | 74.001 |
| Java GZip | - | 21.117 | 1 | 0.283 | 74.796 |
| Apache Deflate | - | 21.064 | 1 | 0.276 | 72.841 |
| Apache Deflate | - | 21.041 | 2 | 0.553 | 72.851 |
| Java Zip | - | 21.036 | 4 | 1.116 | 73.375 |
| ZLib - Default | 3 | 21.023 | 8 | 2.216 | 72.807 |
| ZLib - Huffman | 5 | 21.022 | 8 | 2.216 | 72.807 |
| ZLib - Default | 1 | 21.016 | 8 | 2.217 | 72.807 |
| ZLib - Default | 2 | 21.005 | 8 | 2.218 | 72.807 |
| ZLib - Huffman | 3 | 21.002 | 8 | 2.218 | 72.807 |
| ZLib - Huffman | 4 | 21.002 | 8 | 2.218 | 72.807 |
| ZLib - Filtered | 4 | 21.000 | 8 | 2.218 | 72.807 |
| ZLib - Filtered | 2 | 20.999 | 8 | 2.218 | 72.807 |
| ZLib - Filtered | 3 | 20.998 | 8 | 2.219 | 72.807 |
| Apache GZip | - | 20.995 | 1 | 0.277 | 72.839 |
| ZLib - Default | 4 | 20.991 | 8 | 2.219 | 72.807 |
| Java GZip | - | 20.991 | 2 | 0.564 | 74.004 |
| ZLib - Default | 5 | 20.987 | 8 | 2.220 | 72.807 |
| ZLib - Huffman | 2 | 20.976 | 8 | 2.221 | 72.807 |
| ZLib - Huffman | 1 | 20.974 | 8 | 2.221 | 72.807 |
| Java GZip | - | 20.973 | 4 | 1.119 | 73.376 |
| ZLib - Filtered | 1 | 20.941 | 8 | 2.225 | 72.807 |
| Apache GZip | - | 20.923 | 2 | 0.557 | 72.850 |
| ZLib - Filtered | 5 | 20.865 | 8 | 2.233 | 72.807 |
| ZLib - Default | 3 | 20.793 | 4 | 1.120 | 72.839 |
| ZLib - Default | 1 | 20.785 | 4 | 1.121 | 72.839 |
| ZLib - Default | 5 | 20.774 | 4 | 1.121 | 72.839 |
| ZLib - Filtered | 3 | 20.772 | 4 | 1.122 | 72.839 |
| ZLib - Filtered | 5 | 20.771 | 4 | 1.122 | 72.839 |

Tabela I.1: (Continuação)

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| ZLib - Filtered | 1 | 20.768 | 4 | 1.122 | 72.839 |
| ZLib - Default | 4 | 20.762 | 4 | 1.122 | 72.839 |
| ZLib - Default | 2 | 20.760 | 4 | 1.122 | 72.839 |
| ZLib - Huffman | 1 | 20.757 | 4 | 1.122 | 72.839 |
| ZLib - Filtered | 2 | 20.751 | 4 | 1.123 | 72.839 |
| ZLib - Huffman | 2 | 20.745 | 4 | 1.123 | 72.839 |
| ZLib - Huffman | 4 | 20.744 | 4 | 1.123 | 72.839 |
| ZLib - Huffman | 3 | 20.740 | 4 | 1.123 | 72.839 |
| ZLib - Filtered | 4 | 20.720 | 4 | 1.124 | 72.839 |
| ZLib - Huffman | 5 | 20.680 | 4 | 1.127 | 72.839 |
| ZLib - Default | 2 | 20.635 | 2 | 0.564 | 72.851 |
| ZLib - Default | 5 | 20.634 | 2 | 0.564 | 72.851 |
| ZLib - Default | 1 | 20.634 | 2 | 0.564 | 72.851 |
| ZLib - Filtered | 1 | 20.630 | 2 | 0.565 | 72.851 |
| ZLib - Filtered | 2 | 20.629 | 2 | 0.565 | 72.851 |
| ZLib - Default | 4 | 20.615 | 2 | 0.565 | 72.851 |
| ZLib - Default | 3 | 20.610 | 2 | 0.565 | 72.851 |
| ZLib - Default | 4 | 20.557 | 1 | 0.283 | 72.841 |
| ZLib - Huffman | 2 | 20.524 | 2 | 0.567 | 72.851 |
| ZLib - Filtered | 3 | 20.517 | 2 | 0.568 | 72.851 |
| ZLib - Huffman | 3 | 20.503 | 2 | 0.568 | 72.851 |
| ZLib - Filtered | 1 | 20.495 | 1 | 0.284 | 72.841 |
| ZLib - Filtered | 2 | 20.495 | 1 | 0.284 | 72.841 |
| ZLib - Default | 3 | 20.483 | 1 | 0.284 | 72.841 |
| ZLib - Huffman | 5 | 20.464 | 1 | 0.284 | 72.841 |
| ZLib - Filtered | 4 | 20.461 | 2 | 0.569 | 72.851 |
| ZLib - Filtered | 3 | 20.405 | 1 | 0.285 | 72.841 |
| ZLib - Default | 5 | 20.395 | 1 | 0.285 | 72.841 |
| ZLib - Filtered | 4 | 20.386 | 1 | 0.285 | 72.841 |
| ZLib - Filtered | 5 | 20.368 | 2 | 0.572 | 72.851 |
| ZLib - Default | 1 | 20.363 | 1 | 0.286 | 72.841 |
| ZLib - Filtered | 5 | 20.357 | 1 | 0.286 | 72.841 |
| ZLib - Huffman | 1 | 20.352 | 1 | 0.286 | 72.841 |
| ZLib - Huffman | 2 | 20.347 | 1 | 0.286 | 72.841 |
| Apache GZip | - | 20.324 | 4 | 1.146 | 72.839 |
| ZLib - Default | 2 | 20.323 | 1 | 0.286 | 72.841 |
| ZLib - Huffman | 1 | 20.323 | 2 | 0.573 | 72.851 |
| ZLib - Huffman | 4 | 20.272 | 2 | 0.574 | 72.851 |
| ZLib - Huffman | 3 | 20.269 | 1 | 0.287 | 72.841 |
| ZLib - Huffman | 4 | 20.205 | 1 | 0.288 | 72.841 |
| ZLib - Huffman | 5 | 19.648 | 2 | 0.593 | 72.851 |
| Apache BZip | - | 8.547 | 128 | 91.370 | 76.269 |
| Apache BZip | - | 6.848 | 64 | 56.608 | 75.716 |

Tabela I.1: (Continuação)

| Compressor | Nível | ratio/time | Tamanho | Tempo Médio (s) | Ratio Médio (%) |
|---|---|---|---|---|---|
| Apache BZip | - | 5.956 | 32 | 32.391 | 75.369 |
| Apache BZip | - | 5.766 | 16 | 16.764 | 75.520 |
| Apache BZip | - | 5.545 | 8 | 8.717 | 75.542 |
| Apache BZip | - | 5.465 | 1 | 1.105 | 75.506 |
| Apache BZip | - | 5.424 | 4 | 4.457 | 75.556 |
| Apache BZip | - | 5.389 | 2 | 2.242 | 75.548 |
| Apache XZ | - | 4.320 | 128 | 188.520 | 79.542 |
| Apache XZ | - | 4.312 | 1 | 1.425 | 76.823 |
| Apache XZ | - | 3.770 | 2 | 3.274 | 77.172 |
| Apache XZ | - | 3.457 | 64 | 116.472 | 78.648 |
| Apache XZ | - | 3.332 | 4 | 7.436 | 77.448 |
| Apache XZ | - | 3.112 | 8 | 15.984 | 77.725 |
| Apache XZ | - | 3.075 | 16 | 32.437 | 77.937 |
| Apache XZ | - | 3.059 | 32 | 65.292 | 78.027 |