UMA ARQUITETURA DE SOFTWARE PARA APLICAÇÕES DE LARGA ESCALA E DE TEMPO REAL

Miguel Kassick Cadaviz[1]

Kleinner Silva Farias de Oliveira[2]

Resumo: Com a popularização da internet e o aumento do volume de dados que circulam por aplicações coorporativas que exigem que grandes volumes de informações sejam processados em tempo real, a indústria de *software* tem disponibilizado vários *frameworks* com o intuito de prover o serviço necessário. Contudo, a cada dia que passa os dados estão atingindo uma escala cada vez maior, tornando as arquiteturas sugeridas pela literatura obsoletas e limitadas, não conseguindo suportar a demanda e o grande volume de elementos em larga escala e em tempo real. Os modelos adotados atualmente frequentemente são monolíticos e inflexíveis, dificultando sua implementação e qualquer alteração de escopo necessária. Portanto, o trabalho propõe um modelo arquitetural visando dar suporte a aplicações que trabalhem nesse contexto, dando à devida atenção a aspectos como reaproveitamento, escalabilidade e padronização.

Palavras-chave: tempo real; larga escala.

1 INTRODUÇÃO

Com o crescimento surpreendente da internet e da internet das coisas, tem-se uma vasta quantidade de informações disponíveis online, onde diversas aplicações criam enormes quantidades de informações estruturadas e não estruturadas, que precisam ser processadas, analisadas, vinculadas e armazenadas. O tempo para processá-las é demasiadamente curto e o grande volume de dados versus o curto espaço de tempo que os usuários estão dispostos a esperar até receber algum tipo de resposta, apresenta o maior dos desafios (PUIU, 2016).

[1] Estudante de Ciência da Computação na Universidade do Vale do Rio dos Sinos – UNISINOS, São Leopoldo, Brasil. E-mail: mkcadaviz@edu.unisinos.br.
[2] Professor assistente do Programa de Pós-Graduação em Computação Aplicada (PIPCA) da UNISINOS. Doutor em Informática pela Pontifícia Universidade Católica do Rio de Janeiro – PUC - Rio, Rio de Janeiro, Brasil. E-mail: kleinnerfarias@unisinos.br.

Em um cenário como este, é necessário aliar abordagens da Real-time Computing (RTC) com métodos da Large-Scale System (LSS) para que seja possível atender as expectativas de maneira satisfatória. No contexto acadêmico da computação, a RTC descreve sistemas de hardware ou software que atuam sujeitos a limitação de tempo. Esta restrição, embora seja variável, costuma estar na ordem de milissegundos e, algumas vezes, em microssegundos (KUO et al.,2013). Um sistema de tempo real pode ser definido como aquele que controla um ambiente pela recepção dos dados, seu processamento e o retorno dos resultados de maneira suficientemente rápida, para que seja capaz de influenciar no ambiente na mesma porção de tempo. O termo “tempo real”, em alguns casos, também pode indicar que o sistema em questão não apresenta nenhum atraso significativo. O termo Large-Scale System é utilizado para se referir a sistemas com um grande volume de dados, de usuários ou de hardware (DUBITZKY et al., 2012). Ao se trabalhar com sistemas em larga escala, os métodos tradicionais de engenharia e gerenciamento de software não são mais adequados e devem ser revistos, segundo (NORTHROP, 2006).

Aplicações em larga escala e de tempo real são utilizadas no cotidiano para a resolução de problemas de forma pontual e desnormalizada, porém ainda se estuda arquiteturas e se definem modelos para sua construção. Essa falta de padronização impacta no projeto de novos sistemas pois, na falta de uma arquitetura abrangente ao novo problema, constrói-se uma totalmente nova. Também ocorrem problemas quando existe a necessidade de alterar componentes ou o funcionamento de uma ou mais etapas do processo, pois este costuma ser elaborado de forma acoplada; sendo assim, a alteração em um ponto da estrutura acaba impactando o conjunto como um todo. Por fim, existe uma curva de aprendizado a ser percorrida quando é feita a transição de um sistema para o outro. Neste meio, diferentes modelos e abordagens são amplamente discutidos e nota-se que aqueles que propõem uma estrutura modularizada têm obtido uma aceitação satisfatória, visto que esta abordagem facilita a configuração e manutenção de cada etapa.

O objetivo deste trabalho é desenvolver uma arquitetura aderente ao processo de processamento em larga escala e de tempo real. Para alcançar tal propósito foi realizado um estudo sobre ferramentas e sistemas que suportem aplicações em larga escala de tempo real e propõem-se uma arquitetura modular capaz de atender as metas definidas.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os conceitos que contribuem para o bom entendimento do texto. A Seção 3 apresenta alguns trabalhos relacionados. A Seção 4 apresenta a arquitetura proposta. A Seção 5 descreve os resultados da avaliação empírica e a Seção 6 apresenta as conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção são descritos os principais conceitos que ajudarão na compreensão do conteúdo a ser apresentado ao longo deste artigo. A Seção 2.1 introduz o modelo tradicionalmente explorado para soluções de sistemas em larga escala e de tempo real e os problemas inerentes a sua utilização. A Seção 2.2 descreve a definição de fonte de dados, enquanto a Seção 2.3 traz uma visão da camada de publicação e recebimento. O processamento de informações é analisado na Seção 2.4 e a análise dos dados é tratada na Seção 2.5. Por fim, a Seção 2.6 detalha a etapa de armazenamento de elementos e a Seção 2.7 especifica o papel dos consumidores.

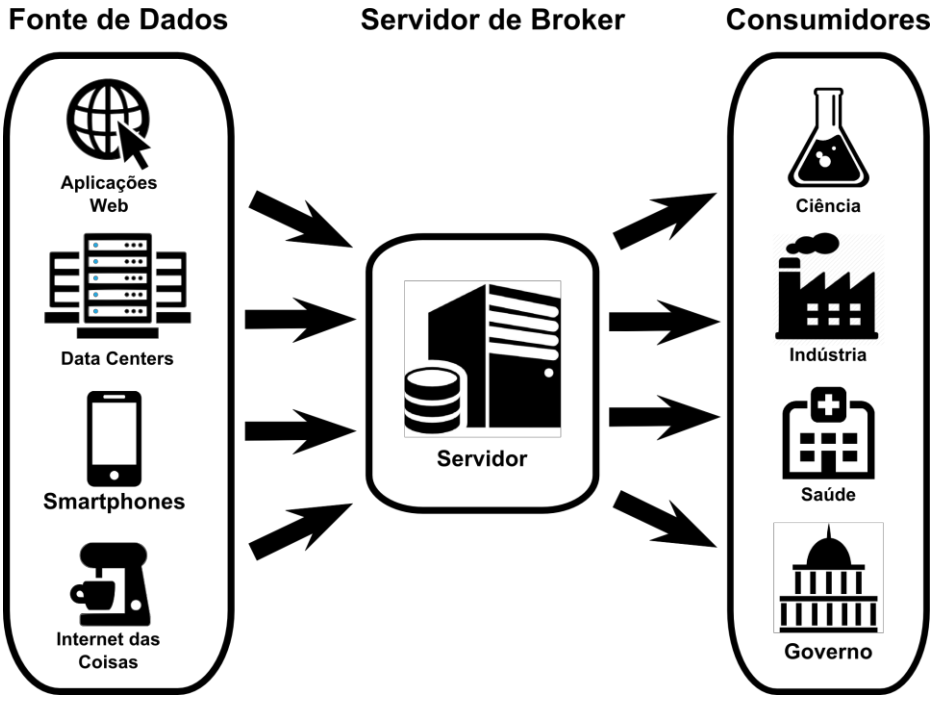
2.1 Modelo Tradicional de Sistemas em Larga Escala e Tempo Real

O processamento de grandes volumes de dados em curto período de tempo já é uma atividade recorrente em várias áreas da computação atual, inclusive em aplicações que julgadas como triviais, como redes sociais, por exemplo, podem trafegar e computar um imenso volume de informações. O modelo atualmente empregado atende às necessidades de uma forma geral, porém apresenta algumas limitações. A Figura 1 mostra a estrutura normalmente utilizada na implementação deste tipo de serviço. Três áreas distintas: (1) as fontes de dados; (2) o servidor de *broker;* (3) e os consumidores.

O conceito de Fonte de Dados, que será descrito mais adiante, trabalha com *streams* de dados que podem ser geradas de diferentes maneiras, as quais posteriormente são encaminhadas a um servidor que atua como *broker.* É possível a utilização de fontes de dados distintas na construção de um sistema de processamento de larga escala e de tempo real. Os Consumidores, como o nome sugere, são sistemas independentes que utilizam as informações processadas e

geradas pela etapa anterior para execução de suas tarefas. Em alguns sistemas, conhecidos como retroalimentados, os consumidores podem também atuar como fontes de dados.

Figura 1 – Arquitetura tradicional



Fonte: elaborado pelo autor.

Um servidor de *Broker* é projetado para gerenciar o envio e recebimento de mensagens, como se este conversasse com as aplicações. A utilização dessa ferramenta permite a concepção de sistemas que trafeguem mensagens de maneira assíncrona e livre de acoplamentos, uma vez que a aplicação lida apenas com as mensagens e não precisa conhecer o sistema que as recebe ou envia. Neste servidor, além do serviço de mensageria, é implementado também um serviço de processamento, análise e armazenamento das informações trafegadas e das oriundas do processamento.

Porém, desta forma, o servidor acaba concentrando várias atividades diferentes e normalmente existe um forte acoplamento entre suas implementações. Apesar de não haver nenhum prejuízo na comunicação da arquitetura com suas fontes de dados e seus consumidores, de desempenho ou qualquer outro problema de funcionamento propriamente dito, a ausência da separação de responsabilidades

e dos sistemas podem acarretar algumas vulnerabilidades, como: (1) dificuldade na substituição de sistemas que componham as etapas de transporte de mensagens, processamento, análise e armazenamento de dados; (2) a ausência de uma arquitetura distribuída implica na fragilidade de um ponto único de falha para todo o sistema, uma vez que, parando o servidor de *broker,* paramos toda a estrutura; (3) a centralização de inúmeras tarefas em um mesmo componente pode acarretar problemas de performance em determinadas áreas sensíveis, sobrecarregando-as enquanto outras seções estão atuando abaixo de sua capacidade máxima; (4) expansão do modelo ou alteração de requisitos primários; (5) uma grande curva de aprendizado a ser percorrida para qualquer interação necessária com o modelo. Além disso, dado o grande acoplamento e a sobreposição das soluções utilizadas, a arquitetura acaba se tornando descartável, uma vez que não pode ser reaproveitada parcial ou totalmente para a resolução de outro problema.

Baseado na análise do modelo tradicional é possível definir seis papéis distintos presentes em uma arquitetura de processamento em larga escala de tempo real: (1) fonte de dados; (2) publicação e recebimento; (3) processamento de informações; (4) análise de dados; (5) armazenamento dos elementos; e (6) consumidores. A modularização e interdependência das camadas é o objetivo principal dessa proposta, aspectos como comportamento e forma de atuação de cada nível serão discutidos nas próximas seções.

2.2 Fonte de Dados

A camada de Fonte de Dados, como dito anteriormente, engloba uma grande diversidade de dispositivos, tais como sensores, maquinário industrial, *data centers,* aplicações web e qualquer outro equipamento que utilize a internet das coisas (STRATA, 2015). Segundo (EVANS, 2011), a internet mudará o mundo e também todos seus habitantes. Naquela época, talvez, ainda houvesse alguma dúvida, porém hoje em dia é inegável o impacto que a internet tem em áreas como educação, ciência, negócios, além de outras do cotidiano. Pessoas estão conectadas à internet o tempo todo, inclusive quando acham que não estão e é nesse contexto que surge o conceito de Internet das Coisas (IoT), que é a evolução da internet como se conhece, pois possui a capacidade de coletar, analisar e

distribuir dados que podem ser transformados em vários tipos distintos de conhecimento (SINGH, 2014).

Em 2003, a população mundial era de, aproximadamente, 6,3 bilhões de pessoas e havia 500 milhões de dispositivos conectados à internet. A razão entre o número de dispositivos conectados e a população mundial é de 0,08, um número bastante pequeno. O crescimento expressivo da tecnologia móvel eleva essa relação a 1,84 em 2010 (EVANS, 2011). No ano de 2016, a quantidade de dispositivos conectados por pessoa é de 3,13 se considerarmos todas as pessoas do mundo, ou 6,36 considerando apenas a fatia que possui acesso à internet. De acordo com (STATISTA, 2016), o número de dispositivos com acesso à internet deve chegar a 50,1 bilhões até o ano de 2020, estimativa bem semelhante à realizada pelo Cisco IBSG presente em (EVANS, 2011).

Fica evidente a grande diversificação de informações que a utilização de diferentes dispositivos nos possibilita. Além disso, existe a enorme velocidade com a qual a tecnologia vem se desenvolvendo e fornecendo novas formas de gerar informação. A arquitetura deve ser expansível de tal maneira que apresente uma interface de comunicação que seja capaz de suportar diferentes fontes de dados e também comportar novos padrões que podem ser criados no futuro.

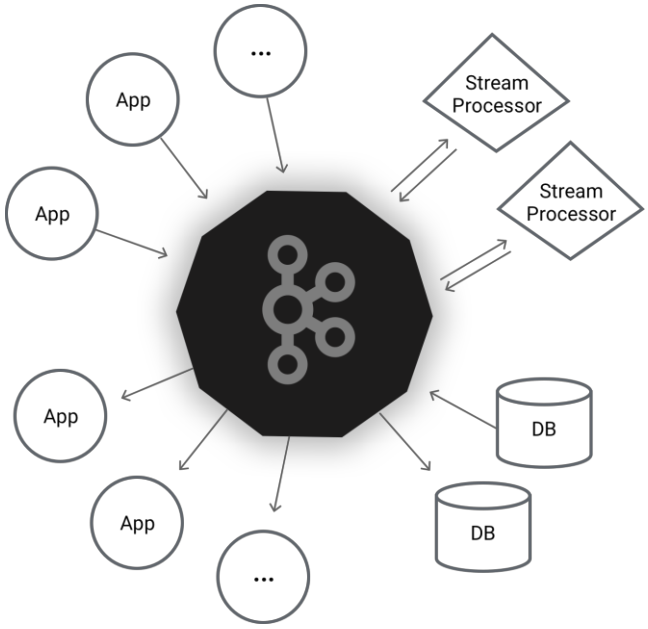
2.3 Publicação e Recebimento

Esta camada é comumente mencionada em inglês por *pub/sub.* Neste nível da arquitetura, as mensagens provenientes das diversas fontes de dados são publicadas em outro ambiente, normalmente um *cluster*, para que possam ser processadas em alta velocidade. Essa área do sistema também permite que aplicações se inscrevam para receber informações já processadas, em outras palavras, age como o sistema nervoso central, coletando informações em alto volume e disponibilizando-as para o consumo de outras aplicações. Em comparação com o modelo tradicional, é neste nível que existe o serviço de *broker* de mensagens, intermediando as comunicações entre os demais componentes da arquitetura.

O Apache Kafka é um agente de mensagens de código aberto. A Figura 2 apresenta uma sugestão de estrutura para implementação, na qual é possível observar a presença dos papéis descritos anteriormente (KAFKA, 2016). No modelo,

existem aplicações que operam como fonte de dados, enquanto outros consomem as informações geradas. Verifica-se que todo o fluxo de informações passa pelo *Broker*, que realiza o controle de dados entre os serviços que processam as *streams*, as bases de dados, as fontes de dados e os consumidores.

Figura 2 – Estrutura do *broker* do Kafka



Fonte: (KAFKA, 2016).

2.4 Processamento de Informações

Este componente prevê o processamento das *streams* de dados que foram geradas na primeira camada e transportadas pela segunda. O processamento de *streams* é um paradigma que permite algumas aplicações explorarem mais facilmente modelos de processamento paralelo, simplificando implementações de *software* e *hardware* (BEARD, 2015). Esta etapa normalmente ocorre de forma paralelizada em um cluster, assim como a primeira etapa, a fim de potencializar a velocidade de processamento e tráfego das informações.

2.5 Análise de Dados

A etapa de análise é responsável por examinar os dados coletados, mapeando possíveis relacionamentos entre as informações, cruzando vários tipos de registros, avaliando possíveis conseqüências em suas interações e, por fim, armazenando os resultados obtidos, a fim de permitir uma leitura rápida e otimizada

de fácil compreensão por aqueles que vierem a consumir estes resultados nas camadas subsequentes.

A análise é realizada por aplicações disparadas pela etapa de Processamento de Informações que recebem porções de informações. São iniciadas, de forma paralela, várias instâncias dessas aplicações, que recebem diferentes blocos de informações, para que o processamento se dê de forma rápida, simultânea e tolerante a falhas. Este método favorece o desacoplamento da área de análise do resto da arquitetura, uma vez que esta apenas recebe informações, as processa e retorna o resultado obtido, desconhecendo completamente o restante da estrutura.

2.6 Armazenamento

Após realizada a interpretação dos dados é necessário que estes sejam gravados para que estejam disponíveis aos consumidores. Normalmente, são implementados bancos de dados não relacionais, pois possuem afinidade a grandes volumes de informações e a execução de serviços escalonáveis, ou a computação em nuvem, onde existe uma necessidade de uma escalabilidade horizontal, podendo inclusive trabalhar em conjunto com bancos de dados relacionais. Já os bancos de dados relacionais nem sempre são a melhor opção em cenários onde existe a necessidade de armazenar estruturas dinâmicas, tratar grandes volumes de dados ou lidar com estruturas não convencionais como grafos.

2.7 Consumidores

Após serem processadas e analisadas, as informações estão prontas para serem utilizadas em uma área de interesse específica e é nesse estágio que encontramos os consumidores. Dados financeiros e padrões de consumo podem ser cruzados e utilizados em serviços financeiros e de marketing; pode-se melhorar o serviço de saúde utilizando o prontuário médico de um paciente e comparando-o com o de seus familiares; entre muitas outras possibilidades.

Segundo (PUIU, 2016), explorar os avanços da internet das coisas para incentivar a criação de cidades inteligentes propõe atacar, desde desafios urbanos comuns como redução do consumo de energia ou o congestionamento das principais vias e a poluição do ambiente, até desafios mais complexos, como

problemas de infraestrutura urbana, de segurança e de distribuição de renda, por exemplo.

3 TRABALHOS RELACIONADOS

Existe uma preocupação crescente com os modelos e configurações de arquitetura mais aderentes para cada modelo de implementação que viabilizam um grande volume de dados em tempo real. A literatura recente tem explorado bastante esse tema, dessa forma esta Seção apresenta a situação atual deste tema, explorando diversas pesquisas e uma comparação com a pesquisa proposta.

3.1 Processamento em larga escala e de tempo real para aplicações científicas

Baseado no conceito de DDDAS (Dynamic Data-Driven Application Systems), (CAO; LI, 2011) propõem um *framework* com objetivo de otimizar o desempenho do processamento de dados do LIGO, onde dados são gerados em alto volume, cerca de 10 MB por segundo, quase 1 TB por dia. Outro desafio é a questão de transporte, uma vez que os equipamentos estão distribuídos por milhares de quilômetros de distância. Neste contexto o desempenho é um fator crítico, uma vez que o processamento das informações colhidas deve ser concluído dentro de 30 minutos.

A arquitetura proposta pelo trabalho baseia-se em quatro camadas: (1) *aplicação:* permite a entrada de parâmetros e publica os resultados de maneira amigável ao usuário final; (2) *processamento de dados:* realiza o processamento de dados adicionais baseando-se nas saídas do monitoramento; (3) *monitoramento de dados:* acompanha dinamicamente as informações de medição em um monitor central; e (4) *instrumentação:* que recebe elementos coletados de um ou mais sistemas de medição e os submete a etapas de tratamento, posteriormente armazenando o conjunto na camada de instrumentação. Segundo (CAO; LI, 2011) os pontos fortes do *framework*, estão nos algoritmos de redução de dados, capazes de reduzir um bloco de dados de 9063 MB para 29 MB, reduzindo assim possíveis gargalos na área de transporte.

O autor acredita que, em um futuro próximo, as escalas de dados crescerão exponencialmente. Neste cenário, apesar do conceito de DDDAS ser direcionado

principalmente para aplicações de simulação, o *framework* terá a possibilidade de revelar seu potencial para diversas aplicações cientificas, além do LIGO.

3.2 Processamento em tempo real para estações meteorológicas

Na China, é comum a utilização de estações meteorológicas autônomas, as quais somam mais de 30 mil espalhadas pelo país (WU et al., 2011). Os dispositivos de aquisição dos dados e terminais de controle estão comumente localizados em diferentes locais, tais como: campos, ilhas, reservas, dentre outros.

Utilizando o conceito de cliente/servidor, todos os dados coletados são enviados para um servidor de dados simultaneamente. Neste sistema, onde existe uma grande rede de transmissão de dados que são transmitidos com uma pequena quantidade de informação e em larga escala, fazendo uso de transmissões sequenciais, se faz necessária a implementação de um processamento em tempo real eficiente, capaz de receber e tratar os dados de forma síncrona.

O trabalho apresentado em (WU et al., 2011) expõe uma discussão detalhada sobre o projeto de um sistema para recebimento e processamento de dados das estações meteorológicas autônomas da província de Guangdong. Cada posto de observação produz um grupo de dados a cada 5 minutos em dias de tempo bom ou 1 minuto em situações atípicas. Dessa forma, o modelo deve ser auditável e extensível conforme as necessidades de técnicas ou de operação das estações meteorológicas, além de evitar o congestionamento no *upload* de dados de observação em momentos críticos. Para isso, a premissa é que todo o ciclo de processamento e armazenamento de dados deve ser concluído no período de 1 minuto ou menos.

O sistema foi colocado em produção e aplicado em mais de 1700 estações meteorológicas por mais de um ano. Funcionando com estabilidade, eficiência e confiança. Todas as estações enviam seus pacotes à central de controle ao mesmo tempo a cada minuto. O tempo de recebimento, processamento e *upload* da resposta para cada ciclo ficou abaixo de 1 minuto e os requisitos de *hardware* são baixos, barateando a manutenção e reduzindo a complexidade dos sistemas utilizados.

3.3 Escalonamento em tempo real para Controle de Aplicações em Larga Escala

Este trabalho defende que os requisitos de aplicações em tempo real podem ser satisfeitos por um esquema de escalonamento adequado a um ambiente de programação baseado em modelos. Pesquisadores de controle de processos da Universidade de Vanderbilt desenvolveram um IPCS (*Intelligent Process Control System*), que é um sistema baseado em modelos e age como um supervisor em tempo real de processos industriais de larga escala, capaz de controlar várias atividades, monitorar diferentes operações e fazendo uso das informações de sensores, é capaz de diagnosticar uma falha no sistema (WAKNIS; SZTIPANOVITS, 1993).

O esquema é construído em três camadas: (1) *camada de ferramentas de construção de modelos*, que permite a representação qualitativa e quantitativa da informação; (2) *camada de interpretação de modelos*, onde é armazenada a informação de forma descritiva, que é utilizada para monitoramento, controle e diagnóstico, além de ter a capacidade de transformar os modelos em estruturas computacionais executáveis; e (3) a *camada de execução do ambiente*, que é implementada com a arquitetura multigrafo, utilizando grafos direcionais. Os nós do grafo representam os blocos computacionais e suas conexões representam o fluxo de dados entre eles. Cada bloco possui um segmento de programa que são escalonados conforme a demanda e o fluxo de dados.

Uma característica notável que é oferecida por essa estrutura é a capacidade de reconfiguração dinâmica. Esta não é limitada apenas a ajustes nos parâmetros do sistema, permitindo mudanças arquiteturais em tempo de execução pela inclusão e remoção de nós e conexões. A estrutura também apresenta interfaces para o recebimento de informações do ambiente externo e integração com seus consumidores; além disso, também possui uma interface, implementada por cada nó, responsável pelo processamento de forma paralela e escalonável dos dados de entrada e outra de função similar que processa os dados de saída.

3.4 Análise comparativa

Muitos projetos já foram desenvolvidos na área, visando obter soluções que procuram atender problemas presentes em situações bem particulares. A comparação entre os projetos abrange quesitos de vital importância para o modelo proposto, tais como a modularização, a capacidade e facilidade de expansão do modelo aplicado, bem como a escalabilidade da solução implementada. Cada critério descrito anteriormente foi avaliado em separado para cada um dos seis papéis essenciais na arquitetura de processamento em larga escala de tempo real. Os resultados desta comparação podem ser observados na Tabela 1.

Tabela 1 – Análise Comparativa

	Modularizado	Fonte de Dados	Publicação e Recebimento		Processamento de Informações		Análise		Armazenamento		Consumidores
		Expansível	Expansível	Escalonável	Expansível	Escalonável	Expansível	Escalonável	Expansível	Escalonável	Expansível
Processamento em larga escala e de tempo real para aplicações científicas	~	-	~	+	~	+	~	+	~	+	-
Processamento em tempo real para estações meteorológicas	~	-	~	+	~	+	~	+	~	~	+
Escalonamento em tempo real para Controle de Aplicações em Larga Escala	~	~	+	+	~	+	~	+	~	+	+
Arquitetura Proposta	+	+	+	+	+	+	+	+	+	+	+

Legenda: Implementa + Implementa Parcialmente ~ Não Implementa - Não se aplica

Fonte: Elaborado pelo autor

É possível verificar que, apesar da grande escalabilidade dos projetos avaliados, existe uma deficiência na questão de expansão e reutilização da estrutura, por este motivo a arquitetura proposta tem por objetivo, além da satisfação das demandas apresentadas, fornecer uma estrutura dividida em módulos e reaproveitável, fazendo uso de micro serviços e ressaltando a separação entre suas etapas de processamento. Tal implementação favorece a construção de estruturas expansíveis, reaproveitáveis e de fácil compreensão, contribuindo tanto para os sistemas já existentes quanto para os que ainda serão construídos.

4 ARQUITETURA PROPOSTA

A presente seção é dedicada à descrição da arquitetura proposta como solução, bem como a integração entre seus componentes e o funcionamento dos

mesmos. Para isso, a Seção 4.1 apresenta uma visão geral da arquitetura e a Seção 4.2 detalha as tecnologias utilizadas na arquitetura.

4.1 Visão Geral

Nas Seções anteriores foram definidos e debatidos seis papéis essenciais para a consolidação de uma aplicação em larga escala de tempo real. Na (STRATA, 2015) foi apresentado um modelo de cinco áreas para modelagem destes sistemas (ver Tabela 2). Tal proposta visa mapear e modelar uma estrutura em blocos distintos, cada um contendo suas próprias atribuições e responsabilidades, além de facilitar a substituição e configuração de cada etapa, uma vez que estão desacopladas umas das outras e trabalham interfaces em suas entradas e saídas. Também são apresentados exemplos de softwares, equipamentos e áreas de interesse de cada um dos pilares demonstrados. Observando a tabela fica clara a necessidade de mais camadas e sistemas assumindo responsabilidades que sequer existiam em modelos mais simplificados. Por esse motivo, a área responsável por armazenamento e análise foi desmembrada em papéis distintos, a fim de facilitar a implementação e modularização dos mesmos.

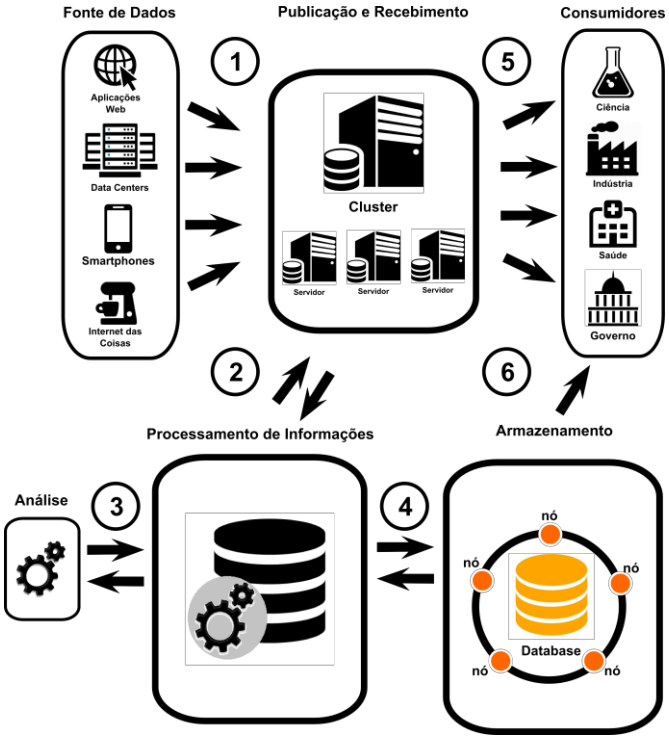
Tabela 2 – Pilares de aplicações em larga escala de tempo real

Fonte de Dados	Publicação e Recebimento	Processamento de Informações	Armazenamento e Análise	Consumidores
Data Centers	Apache Kafka	Apache Spark	Apache Cassandra	Saúde
Aplicações Web	Amazon Kinesis	Apache Storm	Apache HBase	Operações de TI
Imagens Aéreas	Google Pub/Sub	Apache Samza	Apache Druid	Varejo
Máquinas	MapR Streams	Twitter Heron	Elasticsearch	Governos
Edifícios	Apache DistributedLog	Amazon Kinesis	Apache Solr	Agricultura
Veículos		Google Dataflow	MapR-DB	Marketing
Wearables	Azure Event Hubs	Apache Apex	FiloDB	Manufatura
Smartphones		Apache Flink	memsql	Indústrias
Tablets		Apache Flume	Apache Kudu	Ciência
Internet das Coisas		Azure Stream Analytics	Google Cloud Storage	Serviços Financeiros
		Apache Kafka Streams	Google Cloud Bigtable	Agências de Mídia

Fonte: (STRATA, 2015).

A arquitetura desenvolvida tem como objetivo receber dados em larga escala e tempo real de diversos produtores de dados. Estas informações são encaminhadas a um serviço de *broker* que controla as etapas de processamento e análise destes elementos, seu armazenamento e também a disponibilidade dos mesmos a consumidores. Percebe-se que a arquitetura desenvolvida na Figura 3 é muito semelhante à proposta por (KAFKA, 2016) na Figura 2. Além disso, é possível observar que a participação das fontes de dados e dos consumidores não se altera entre o modelo tradicional exposto na Figura 1 e o proposto neste artigo. A grande diferença da arquitetura desenvolvida está na separação das tarefas e da modularização da estrutura.

Figura 3 – Arquitetura desenvolvida



Fonte: elaborado pelo autor.

A Figura 3 apresenta uma visão geral da arquitetura e os principais passos realizados no processamento das informações, descritos a seguir:

1. Os produtores submetem informações para análise;
2. O processamento de dados é realizado com base nas informações recebidas. Após a conclusão desta etapa, as informações poderão ser

reenviadas à etapa de publicação e recebimento para serem armazenadas em um tópico distinto;

3. Caso seja necessário, pode ser disparada uma instância de um programa para a realização de uma análise mais especializada sobre um bloco de informações específicas;
4. Os resultados da análise são armazenados;
5. Os consumidores podem receber as informações sem tratamento;
6. O resultado da análise pode ser consultado pelos consumidores.

4.2 Projeto Arquitetural

O projeto da arquitetura foi realizado em duas etapas. Na primeira, foram identificados os requisitos que devem ser satisfeitos. Na segunda etapa, são descritas as tecnologias que implementaram cada módulo da arquitetura visando suporte aos requisitos definidos. Ambas etapas serão descritas a seguir.

4.2.1 Requisitos da Arquitetura

Os requisitos foram obtidos com base nos resultados de funcionamento esperados, bem como a leitura de relatórios e pesquisas sobre o assunto. Uma breve descrição de cada requisito funcional (RF) e não funcional (RNF) é apresentada a seguir.

RNF01: suportar o recebimento e processamento de informações em larga escala. A arquitetura deve suportar e garantir integridade de informações que trafeguem em grande volume.

RNF02: deve ser capaz de atuar em tempo real. A arquitetura deve ser capaz de apresentar aos seus consumidores as informações analisadas em tempo real.

RNF03: possibilidade de extensão. A arquitetura deve ser construída de modo a possibilitar sua reutilização ou adição de novo módulo.

4.2.2 Arquitetura e Componentes

Visando atender a todos os requisitos detalhados anteriormente e modelar uma arquitetura baseada em componentes, conforme a Figura 3, foram adotadas algumas tecnologias presentes na Tabela 2.

Fonte de Dados. Por se tratarem de componentes externos, as Fontes de Dados não estarão presentes na arquitetura.

Publicação e Recebimento. Para a construção desta etapa foi escolhido o Apache Kafka, que se destaca por possuir código aberto, além de ser amplamente utilizado. Sua implementação é indicada para a construção de dois tipos de aplicações: sistemas de *pipeline* para fluxos de dados em tempo real que, de forma confiável, obtém informações entre sistemas ou aplicações; ou para softwares que, em tempo real, transformem ou reajam a determinado fluxo de dados. Seu sistema promove uma abstração do registro de dados, denominado tópico. Os tópicos são categorias para os quais os registros são publicados. Os tópicos no Kafka são sempre multi-inscrição, isto é, um tópico pode ter *n* consumidores inscritos para acessar seus dados. A versão utilizada do Apache Kafka foi a 0.10.2.0.

Aliado ao Kafka, nesse módulo temos o Apache ZooKeeper que provê diversas funcionalidades para aplicações distribuídas, como, por exemplo: gerenciamento distribuído de configurações, coordenação de informações, travas, entre outras. A versão implementada do ZooKeeper foi a 3.4.9.

Ambos utilizam a versão 8 release 131 do JRE (Java Runtime Environment) que consiste em uma máquina virtual Java para a execução de aplicações.

Processamento de Informações. Dos componentes sugeridos para a implementação dessa etapa verifica-se que vários fazem parte da Licença Apache, uma licença de software livre fornecida pela ASF. Foi escolhido o Apache Spark 2.1.0.

Para a instalação e configuração do Spark é necessária a instalação do Scala, uma das linguagens que o Spark suporta para criação de aplicações. A versão instalada do Scala foi a 2.12.2.

Análise de Dados. Na Tabela 2 não existem sugestões para ferramentas de análise de dados. Isto se deve a capacidade da etapa de Processamento de Dados de permitir análises simples, além da possibilidade de integrar lógicas de análise com a etapa de Armazenamento. Porém, seguindo com os preceitos de flexibilizar a

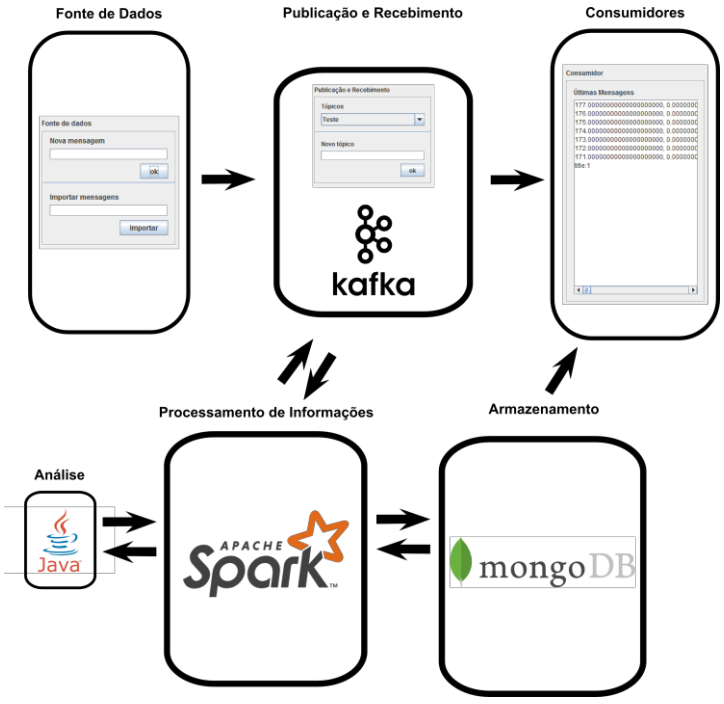
arquitetura, evitar o acoplamento entre as camadas e separar suas responsabilidades, será feita a implementação de aplicações Java para realizar as análises.

Armazenamento de Elementos. Após realizada a análise dos dados se faz necessário que os mesmos sejam gravados para serem disponibilizados aos consumidores. A Tabela 2 cita uma série de bancos de dados não relacionais e voltados a grande volume de dados, porém, foi escolhido o MongoDB, dada a familiaridade do autor com sua utilização. A versão utilizada do MongoDB é a 3.4.4.

Consumidores. Por se tratarem de componentes externos, os Consumidores não estarão presentes na arquitetura.

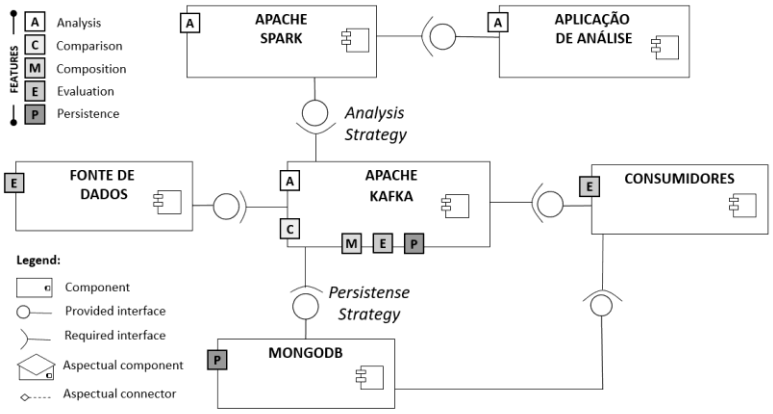
A organização final da arquitetura pode ser observada na Figura 4, onde temos a representação de todos os papéis envolvidos em sistemas de larga escala e tempo real. É importante ressaltar que as funcionalidades referentes à fonte de dados e aos consumidores; e algumas funcionalidades da etapa de publicação e recebimento foram implementadas em um protótipo que será descrito posteriormente. O relacionamento entre os componentes da arquitetura descritos acima pode ser observado no diagrama de componentes da Figura 5.

Figura 4 – Arquitetura Completa



Fonte: elaborado pelo autor.

Figura 5 – Diagrama de Componentes



Fonte: elaborado pelo autor.

5 AVALIAÇÃO DA ARQUITETURA

Esta Seção apresenta uma avaliação empírica da arquitetura. A Seção 5.1 apresenta a metodologia utilizada na avaliação, enquanto a Seção 5.2 descreve os aspectos de implementação da arquitetura. Por fim, a Seção 5.3 apresenta os resultados e inclui melhorias para a arquitetura.

5.1 Método de Avaliação

O propósito da avaliação da arquitetura é apresentar alguns resultados obtidos ao submeter informações à etapa de publicação e recebimento, demonstrando a sua capacidade de lidar com informações em larga escala e tempo real.

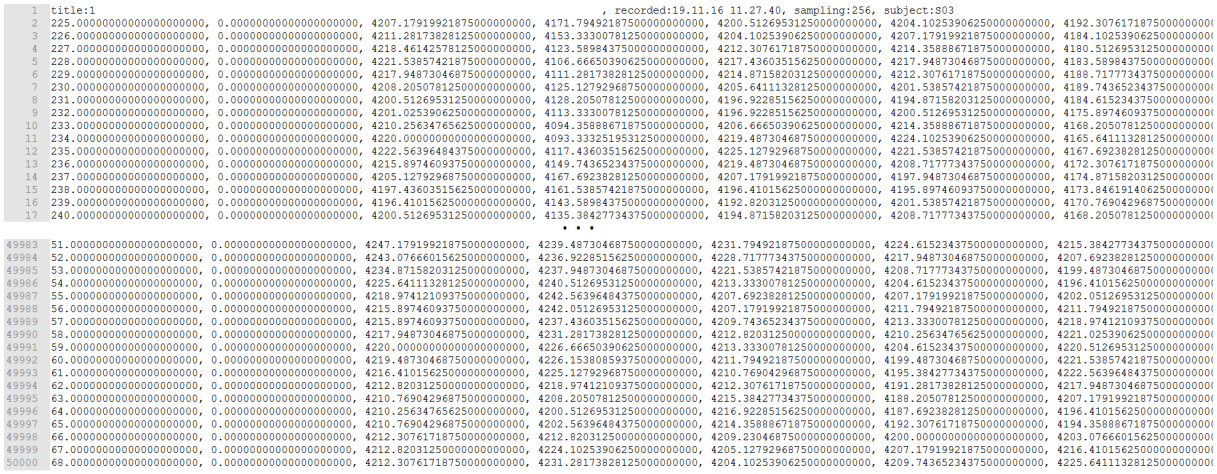
A arquitetura foi construída em uma máquina virtual com os seguintes requisitos:

- Sistema Operacional Windows 10;
- Processador Intel i7 (2,7 GHz) com 4 núcleos disponíveis;
- 4 GB de memória RAM DDR4 de 2133 MHz;
- 30 GB de HD (5400 RPM).

A primeira avaliação considera a capacidade da arquitetura de respostas em tempo real. Para isso, foram submetidos diversos arquivos com informações de um aparelho de eletroencefalograma na extensão csv. O maior deles conta com 50 MB de informações, totalizando mais de 50 mil registros e quase 51 milhões de caracteres. A Figura 6 exibe a estrutura dos arquivos.

A segunda verificação consiste em colocar à prova o conceito de larga escala. Para isso, serão criadas diversas *threads* que estarão associadas ao mesmo produtor e farão publicações de forma simultânea.

Figura 6 – Exemplo arquivo



Fonte: elaborado pelo autor.

5.2 Aspectos de Implementação

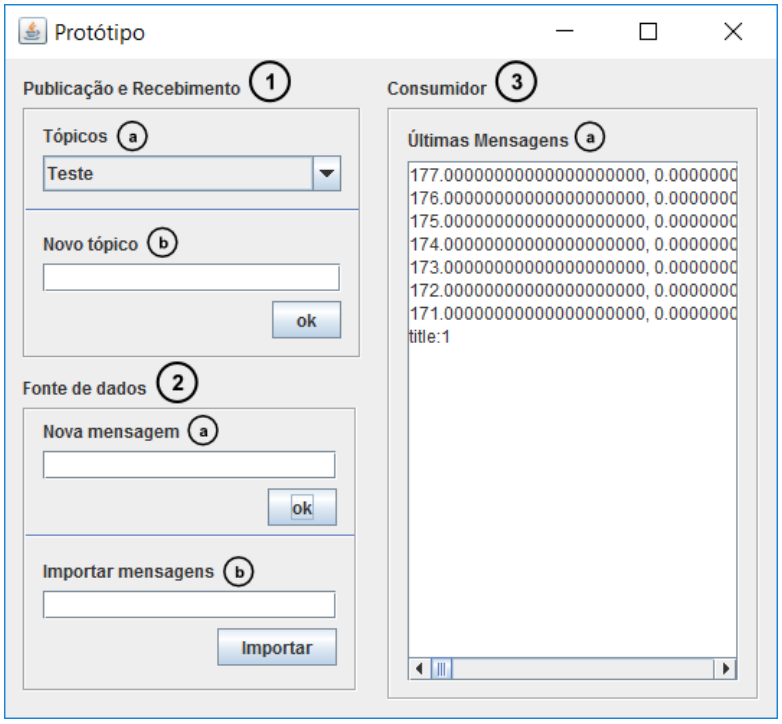
A etapa de publicação e recebimento, implementada com o Apache Kafka, controla todo o fluxo de entrada e saída de dados, atuando como o sistema nervoso central da estrutura, porém, por padrão, as interfaces de produtor e consumidor disponíveis são via *prompt*, utilizando arquivos de lote (extensões bat e sh) e linhas de comando. Além disso, outras atividades cruciais como colocar um servidor Kafka no ar, criar um novo tópico ou até mesmo listar todas as mensagens já recebidas por determinado tópico, seguem este mesmo caminho.

Para facilitar a utilização da arquitetura e criar uma interface para os comandos mais complexos foi construído um protótipo utilizando a linguagem Java e utilizando Maven para fazer o gerenciamento da construção e dependências do projeto. Sua interface foi criada utilizando Swing através das ferramentas visuais da IDE Netbeans. Seu funcionamento se resume a algumas interações simples com o *cluster* do Kafka, além de permitir o envio de informações ao mesmo simulando uma fonte de dados e a obtenção destas informações, como se fosse um consumidor.

A Figura 7 mostra uma visão geral do protótipo e é possível verificar a existência de áreas representando os pilares de aplicações em larga escala de tempo real. Cada uma dessas áreas (destacadas por números), implementa algumas funções (destacadas por letras) inerentes a respectiva área. A seguir será discutida brevemente cada uma dessas opções:

1. **Publicação e Recebimento:** Área destinada às funções básicas desta etapa, como, por exemplo: (a) tópicos: permite alternar entre os tópicos já existentes; (b) novo tópico: permite a criação de um novo tópico.
2. **Fonte de dados:** Área destinada a agrupar as funções de Fontes de Dados. Dentre elas podem ser encontradas: (a) nova mensagem: gera uma mensagem única e avulsa no tópico selecionado; (b) importar mensagens: realiza a importação de mensagens em texto plano do arquivo informado para o tópico selecionado.
3. **Consumidor:** Área destinada a agrupar as funções de Consumidores, entre elas temos: (a) últimas mensagens: exibe as últimas mensagens enviadas ao tópico selecionado.

Figura 7 – Protótipo



Fonte: elaborado pelo autor.

5.3 Resultados

As medições de tempo utilizadas no experimento foram realizadas nas camadas mais superiores do protótipo, criando assim um ambiente mais realista

para o usuário final, porém, tal forma de medição acaba considerando o tempo gasto na transmissão das informações como tempo de processamento.

Foram realizados diversos testes de submissão avulsa de mensagens utilizando o protótipo, os quais obtiveram um tempo médio de resposta de 318 ms. Não foi observada qualquer correlação do tempo de resposta com o tamanho da mensagem, seu conteúdo ou com a recorrência da mesma na estrutura do tópico.

A avaliação da capacidade de respostas em tempo real foi realizada com cinco modelos de arquivos diferentes, os resultados obtidos encontram-se na Tabela 3. O processo de importação foi repetido cinquenta vezes para cada arquivo a fim de obter valores com maior confiabilidade. Nota-se que o tempo necessário para que fossem percorridos e submetidos todos os registros segue uma crescente e sofre um ligeiro aumento com o processamento de arquivos maiores. O tempo médio transcorrido entre o envio do último registro e a notificação de conclusão do processamento da arquitetura no consumidor também foi crescente, mas sofre seu maior aumento para arquivos de 30 MB e após isso, a taxa de aumento diminui novamente. A Figura 8 tem por objetivo ilustrar os resultados demonstrados na Tabela 3.

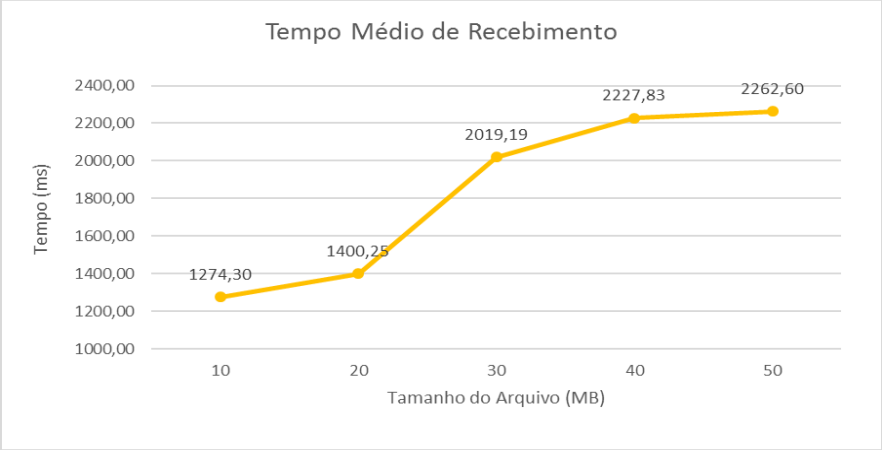
Tabela 3 – Resultados dos testes de Fonte de Dados via arquivo

Tamanho do Arquivo (MB)	Tempo Médio de Envio (ms)	Tempo Médio de Recebimento (ms)
10	343,66	1274,30
20	396,82	1400,25
30	425,62	2019,19
40	906,72	2227,83
50	1628,45	2262,60

Fonte: elaborado pelo autor.

A avaliação da capacidade da arquitetura de lidar com dados em larga escala se deu utilizando diversas *threads* fazendo o papel de fontes de dados simultaneamente utilizando o mesmo tópico. Cada *thread* cria um produtor e gera uma quantidade de mensagens. Foi medido o tempo médio transcorrido durante o envio das mensagens e o tempo médio que a arquitetura precisou para sinalizar que havia recebido todas as informações. A Tabela 4 detalha os resultados obtidos.

Figura 8 – Gráfico com o tempo médio de recebimento



Fonte: elaborado pelo autor.

Tabela 4 – Resultados dos testes de Fonte de Dados via thread

Threads	Número de mensagens	Tempo Médio Envio (ms)	Tempo Médio de Recebimento (ms)
2	2	0	450,83
4	2	0	524,82
8	2	0	671,11
16	2	0	684,27
32	2	3,2	759,04
1024	2	259,2	1820,25
2	20	0	250,02
4	20	0	218,81
8	20	0	331,74
16	20	0	974,86
32	20	6,32	1027,32
1024	20	314,48	2005,72

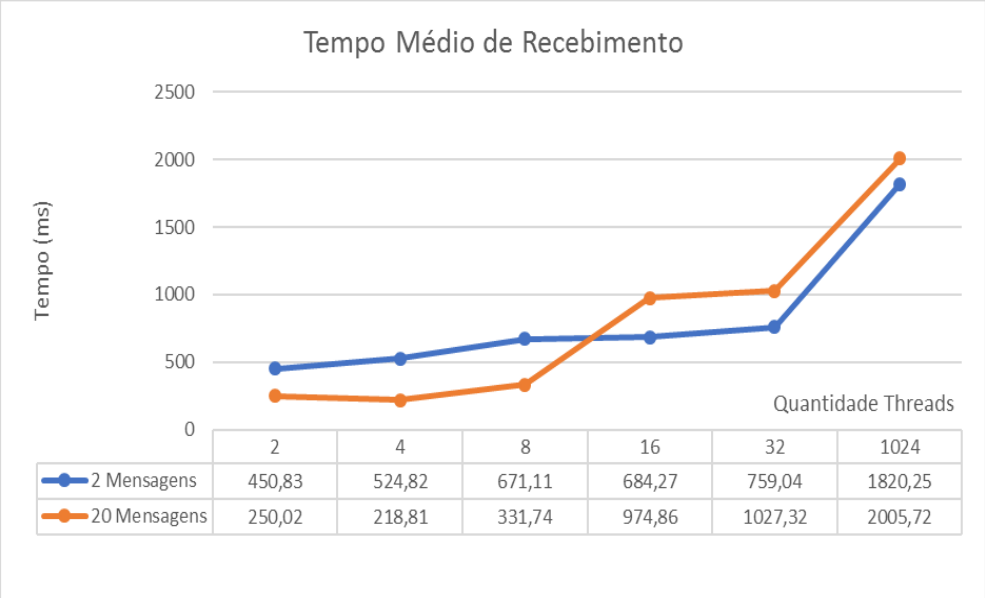
Fonte: elaborado pelo autor.

Nota-se um crescimento gradual no tempo de recebimento com o aumento da quantidade de *threads* e também quanto ao aumento do número de mensagens. Esse comportamento era esperado, porém, os valores encontrados são bem elevados quando comparados com os obtidos na avaliação anterior, entretanto, isso tem um motivo. Como dito anteriormente, os testes foram conduzidos em uma máquina virtual que possui disponibilidade limitada de hardware. Além do mais, os mesmos recursos de hardware são divididos com o sistema na qual a máquina virtual está hospedada. Por este motivo, ao tentar criar uma quantidade maior de *threads,* os processos tendem a ficar na fila de espera, uma vez que não existem

recursos de processador disponível para atender a todos simultaneamente. Este compartilhamento de recursos também é o responsável pela flutuação de valores obtidos como média.

A Figura 9 mostra a representação gráfica dos resultados obtidos. Se considerarmos o caso mais extremo conduzido pelo segundo teste, temos 20 mensagens sendo enviadas por 1024 fontes de dados simultaneamente, totalizando 20.480 mensagens processadas em 2005,72 ms. No contexto da primeira avaliação a quantidade de mensagens foi de 106.650 transmitidas por um único produtor e processadas em 784 ms.

Figura 9 – Gráfico com o tempo médio de recebimento via thread



	2	4	8	16	32	1024
2 Mensagens	450,83	524,82	671,11	684,27	759,04	1820,25
20 Mensagens	250,02	218,81	331,74	974,86	1027,32	2005,72

Fonte: elaborado pelo autor.

6 CONCLUSÃO

A utilização de arquiteturas em tempo real e de larga escala tem ganhado força nos últimos tempos e cada problema apresenta em sua solução uma arquitetura distinta. Essa falta de padronização e modularização dificulta a manutenção e a difusão desse modelo, criando várias estruturas com objetivo similar e construção completamente diferente.

Este artigo apresentou informações sobre arquiteturas em larga escala e de tempo real, suas responsabilidades e seus componentes. Exemplifica também

ferramentas que podem atender cada uma das etapas do processo, além de apresentar o projeto de uma arquitetura flexível e modular.

A avaliação empírica realizada sobre a implementação com o auxílio de um protótipo demonstra que, mesmo utilizando uma modesta configuração de hardware, é possível utilizar a arquitetura para realizar o recebimento, processamento, análise e envio de mensagens em um grande volume num curto espaço de tempo e com bom desempenho. As limitações encontradas na avaliação são relativas ao hardware no qual a arquitetura foi instalada e alguns ajustes na parametrização das etapas visando melhorar o desempenho para o tipo de utilização corrente. Ainda se fazem necessárias algumas melhorias, tais quais: instalação da arquitetura em um ambiente com maior disponibilidade de hardware, realizar testes de configuração de hardware versus desempenho da arquitetura e de configuração dos módulos da arquitetura versus desempenho da mesma.

A SOFTWARE ARCHITECTURE FOR LARGE-SCALE REAL-TIME APPLICATIONS

Abstract: The popularization of the internet and the increasing amount of data circulating through corporate applications that require large volumes of information to be processed in real time, the software industry has made available several frameworks in order to provide the necessary service. However, the data is reaching an everincreasing scale, making the architectures suggested by the literature obsolete and limited, being unable to support this demand and this large volume of large-scale and real-time elements. The currently adopted models are often monolithic and inflexible, hindering their implementation and any required scope change. This paper proposes an architectural model to support large-scale real-time applications, giving due attention to aspects such as reuse, scalability and standardization.

Keywords: real time; large scale

REFERÊNCIAS

APACHE, Confluence (2016). **Clientes Apache Kafka.** Disponível em: <https://cwiki.apache.org/confluence/display/KAFKA/Clients >. Acesso em: 26 de Setembro de 2016.

BEARD, Jonathan (2015). **A Short Intro to Stream Processing.** Disponível em: <http://www.jonathanbeard.io/blog/2015/09/19/streaming-and-dataflow.html>. Acesso em 28 de Outubro de 2016.

CAO, Junwei; LI, Junwei. Large-scale Real-time Data-driven Scientific Applications. In: **2011 Second International Conference on Networking and Distributed Computing**. IEEE, 2011. p. 116-121.

Cassandra, Apache (2016). **Documentação.** Disponível em: <http://cassandra.apache.org/doc/latest/>. Acesso em: 27 de Outubro de 2016.

DUBITZKY, Werner; KUROWSKI, Krzysztof; SCHOTT, Bernard. **Large-scale computing techniques for complex system simulations**. John Wiley & Sons, 2012.

EVANS, Dave. The internet of things. **How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)**, v. 1, p. 1-12, 2011.

INTERNET WORLD STATS (2016). **Internet Usage Statistics. The Internet Big Picture: World Internet Users and Populations Stats.** Disponível em: <http://www.internetworldstats.com/stats.htm>. Acesso em: 10 de Outubro de 2016.

KAFKA, Apache (2016). **Introdução.** Disponível em: <http://kafka.apache.org/intro.html >. Acesso em: 25 de Setembro de 2016.

KUO, Sen M.; LEE, Bob H.; TIAN, Wenshun. **Real-time digital signal processing: fundamentals, implementations and applications**. John Wiley & Sons, 2013.

NORTHROP, Linda et al. **Ultra-large-scale systems: The software challenge of the future**. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2006.

PUIU, Dan et al. CityPulse: Large Scale Data Analytics Framework for Smart Cities. **IEEE Access**, v. 4, p. 1086-1108, 2016.

RABL, Tilmann et al. Solving big data challenges for enterprise application performance management. **Proceedings of the VLDB Endowment**, v. 5, n. 12, p. 1724-1735, 2012.

SINGH, Dhananjay; TRIPATHI, Gaurav; JARA, Antonio J. A survey of Internet-of-Things: Future vision, architecture, challenges and services. In:**Internet of things (WF-IoT), 2014 IEEE world forum on**. IEEE, 2014. p. 287-292.

STATISTA (2016). **The Statistics Portal. Internet of Things (IoT): number of connected devices worldwide from 2012 to 2020 (in billions)**. Disponível em: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide>. Acesso em: 10 de Outrubro de 2016.

STRATA. Large-Scale Real-Time Applications. In: **Strata + Hadoop World Conferences.** 2015

WAKNIS, Prashant; SZTIPANOVITS, Janos. Hard real-time scheduling for large scale process control applications. In: **Real-Time Applications, 1993., Proceedings of the IEEE Workshop on**. IEEE, 1993. p. 71-75.

WOHLIN, Claes et al. **Experimentation in software engineering**. Springer Science & Business Media, 2012.

WU, Guangsheng et al. A Real-time Receiving and Distributed Processing System for Large-scale Burst Data. In: **2011 Second International Conference on Networking and Distributed Computing**. IEEE, 2011. p. 111-115.