CENTRO DOS GUARARAPES CIÊNCIA DA COMPUTAÇÃO
CAMPUS PIEDADE

Beatriz Lira Martins 202002043, Erik Nathan de Oliveira Batista 2020002721, Júlio Cesar de Barros 202002088, Marcos Felipe Vasconcelos de Souza 202004793, Iago Douglas Gonçalves - 202050567

PLATAFORMA HELPINDU: DOCUMENTAÇÃO UTILIZANDO OS SISTEMAS DISTRIBUÍDOS

Recife
2022

CENTRO DOS GUARARAPES CIÊNCIA DA COMPUTAÇÃO
CAMPUS PIEDADE

Beatriz Lira Martins 202002043, Erik Nathan de Oliveira Batista 2020002721, Júlio Cesar de Barros 202002088, Marcos Felipe Vasconcelos de Souza 202004793, Iago Douglas Gonçalves - 202050567

PLATAFORMA HELPINDU: DOCUMENTAÇÃO UTILIZANDO OS SISTEMAS DISTRIBUÍDOS

Trabalho de conclusão de período
da disciplina de Sistemas Distribuídos
apresentando à UNIFG (Faculdade dos Guararapes)
Cumprindo as exigências do curso de
Ciências da computação. Ministrada
pelo professor/mestre João Paulo Santos

Recife
2022

1.Introdução

Um sistema distribuído é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens. Essa definição leva às seguintes características dos sistemas distribuídos: concorrência de componentes, falta de um relógio global e falhas de componentes independentes. Vamos dar três exemplos de sistemas distribuídos: Internet, uma intranet, que é uma parte da internet gerenciada pela organização responsável, computação móvel e ubíqua. Os computadores conectados por meio de uma rede podem estar separados por qualquer distância. Eles podem estar em continentes separados, no mesmo prédio ou na mesma sala. A definição de sistemas distribuídos tem as seguintes consequências importantes:

Concorrência: em uma rede de computadores, a execução concorrente de programas é normal. Posso estar fazendo o meu trabalho em meu computador, enquanto outra pessoa da equipe faz em outro, compartilhando recursos como páginas da web, ou arquivos quando necessário. A capacidade do sistema de manipular recursos compartilhados pode ser ampliada pela adição de mais recursos (por exemplo, computadores) na rede. A coordenação de programas em execução concorrente e que compartilham recursos também é um assunto importante e recorrente.

Falhas independentes: todos os sistemas de computador podem falhar, e é responsabilidade dos projetistas do sistema pensar nas consequências das possíveis falhas. Nos sistemas distribuídos, essas falhas costumam ser diferentes. Falhas na rede resultam no isolamento dos computadores que estão conectados a mesma, mas isso não quer dizer que eles irão parar de funcionar, na verdade, provavelmente os programas não irão conseguir detectar a falha na rede ou se ela ficou mais lenta. Analogamente falando a falha de um computador ou o término inesperado da aplicação (colapso no sistema) não é imediatamente percebida pelos componentes que ali estão tendo uma comunicação. Cada um dos componentes podem chegar a falhar deixando os outros que ali estão ainda em funcionamento. A principal motivação para construir e usar sistemas distribuídos é proveniente do desejo de compartilhar recursos. O termo recurso é bastante abstrato, mas caracteriza bem o conjunto de coisas que podem ser compartilhadas de maneira útil em um sistema de computadores que estão interligados em rede. Arquivos de banco de dados e objetos de dados de todos os tipos . Isso inclui o fluxo de quadros de vídeo, câmera de vídeo digital ou a conexão de áudio que uma chamada de telefone móvel representa.

Os sistemas distribuídos abrangem muitos dos desenvolvimentos tecnológicos mais significativos atualmente, portanto um entendimento da tecnologia subjacente é absolutamente fundamental para o conhecimento da computação moderna. A figura também da vislumbre inicial da ampla variedade de aplicações em uso hoje em dia, contamos com

sistemas de localização relativa, conforme encontrados, em carros ou aviões, até sistemas de escalas globais envolvendo milhares de nós, e também serviços básicos que não necessitam do uso intenso do processador.

2. Objetivos

O HelpIndu é uma plataforma de monitoramento de máquinas industriais online, que tem como objetivo reduzir falhas, performance, custo e tempo em indústrias que trabalham com robôs via IOT. O HelpIndu conta com uma visão completa sobre o trabalho das máquinas industriais reportando falhas a cada setor responsável, reduzindo o tempo de erros ajudando as máquinas a ter uma performance mais alta. Nossa plataforma conta com um dashboard completo de tudo que acontece com as máquinas, cada estágio que elas passam no processo de construção dos produtos de cada setor, fazendo com que a indústria possa ter resultados mais rápido e mais lucratividade. O HelpIndu por ser um grande aliado do IOT trabalha com uma rede própria (Intranet) impedindo assim que haja qualquer tipo de falha entre os sistemas distribuídos, caso haja alguma falha no sistema, os responsáveis são rapidamente acionados pelo HelpIndu, agilizando todo o processo de manutenção para que não haja perca ou danos na construção dos produtos. Todos os equipamentos são ligados via sensores, ajudando na coleta de dados e envio dos relatórios com mais assertividade. Nossa plataforma também pode trabalhar com integração de sistemas, ligando outros setores da empresa como Contas a pagar, contas a receber, RH ou DP. Essa integração pode acontecer via integração de banco de dados, integração de web service, plataforma de processamento de dados eletrônicos, integração de API.

Todas as funcionalidades da plataforma são enviadas em tempo real para os setores responsáveis, usando a otimização, sensorização e conectividade de alta performance, visibilidade e transparência na coleta dos dados, e sendo um sistema flexível para todos os setores da empresa. A proposta busca uma implementação fácil e básica para não comprometer o desenvolvimento dos produtos da empresa. A plataforma também cuida das boas práticas da segurança digital acompanhando os seus quatro pilares: confidencialidade, integridade dos dados, disponibilidade e autenticidade. O suporte da plataforma funciona 24 horas por dia, sete dias na semana, buscando assim não parar a produção em caso de alguma falha na comunicação. Toda alteração ou reparo feito pelo suporte é gerado um relatório para o setor responsável da indústria, para que o cliente possa acompanhar o que está acontecendo na plataforma ligada a sua empresa, todos esses dados são seguros e estarão disponíveis para o cliente na hora que ele precisar.

Disponibilizamos também arquivos de firmware para configuração dos coletores de dados via IOT, todos os arquivos são disponibilizados em nosso site, onde qualquer pessoa pode acessar e baixar os arquivos para uso no sistema.

3. Referencial teórico

A ideia do projeto foi idealizada para Sistemas distribuídos onde os computadores são ligados em rede, se comunicam e coordenam suas ações passando apenas suas mensagens. São usados para aplicações modernas como pesquisas na web, sistemas financeiros ou jogos online. A ideia foi pensada através do estudo do livro:

Sistemas Distribuídos, 5ª Edição de George Coulouris; Jean Dollimore; Tim Kindberg; Gordon Blair

4. Metodologia

Os dados desta pesquisa foram buscados no livro referência de Sistemas Distribuídos, versão atualizada, que pode ser encontrado em nossa biblioteca digital. Os termos para este artigo estão embasados em dados coletados no livro, ajudando-nos assim a ter uma referência de estudo fatídica para o projeto apresentado acima. Com os dados obtidos nesta pesquisa, podemos desenvolver com precisão a plataforma "HelpIndu" nós ajudamos a trazer de forma assertiva para o mercado industrial um projeto agregador que possibilita com grande facilidade a coleta de dados em tempo real via IoT e os Sistemas distribuídos, assim, obtemos o resultado esperado. Usamos como resultado também a otimização da plataforma, a conectividade de forma íntegra, onde todos os dados são coletados sem que haja ruídos nas informações, facilitando o trabalho das pessoas envolvidas.

Quando tratamos de invocação remota em sistemas distribuídos, começamos examinando o serviço mais primitivo, a comunicação por requisição-resposta, que fala sobre os aprimoramentos relativamente pequenos nas primitivas de comunicação entre processos. Falaremos sobre técnicas de invocação remota mais importantes para o sistema distribuído:

- Estratégia chamada de procedimento remoto (PRC) estende-se a abstração de programação da chamada de procedimento para ambientes distribuídos, permitindo que o processo chame um procedimento em um nó remoto como se fosse um local.

- A invocação do método (RMI) é semelhante à o (PRC), mas conta com vantagens adicionais ao usar o conceito de programação orientada a objetos em sistemas distribuídos e em estender o conceito referência de objeto para o ambiente distribuído em geral e permitir o uso de referências de objetos como parâmetros em invocações remotas

Protocolos de requisição-resposta essa função é projetada para suportar as trocas de mensagens cliente-servidor típicas, em casos normais essa requisição é feita de forma

síncrona, fazendo com que o processo cliente seja bloqueado até que o servidor mande de volta uma mensagem para o cliente. Isso faz com que essa comunicação se torne confiável, pois a resposta do servidor volta como uma confirmação para o cliente. A comunicação assíncrona via requisição-resposta é uma alternativa útil em situações em que os clientes podem recuperar as respostas posteriormente.

Aqui faremos uma breve descrição das operações em termos ***send e service*** na API Java para datagramas UDP, embora a grande maioria usem a implementação TCP, um protocolo construído sobre datagramas evita sobrecargas desnecessárias associadas ao protocolo TCP em particular:

- As requisições são bem fechadas pois são seguidas por respostas

- O estabelecimento das mensagens envolvem uma conexão extra entre dois pares de mensagens, além do par exigido por uma requisição e uma resposta

- O controle do fluxo se torna redundante, pois as informações trafegadas passam apenas pequenos argumentos e resultados.

O protocolo de requisição-resposta é baseado em um trio de primitivas de comunicação: ***doOperation, getRequest e SendReply.*** Como já citado acima, o protocolo combina pedidos com respostas. Ele será usado para garantir que as entregas sejam feitas corretamente. Se for usado datagramas UDP, as entregas deverão ser fornecidas pelo protocolo de requisição-resposta, o qual pode usar a resposta do servidor como confirmação da mensagem de requisição do cliente.

Falando sobre as três primitivas de comunicação, o método *doOperation* é usado pelos clientes para invocar operações remotas. Seus argumentos especificam o servidor remoto e a operação a ser invocada, junto com eles vem as informações adicionais que são exigidas pela operação. Seu resultado é um vetor constituído por bytes contendo a resposta. É esperado que o cliente que chama *doOperation* empacota o argumento em um vetor de bytes e desempacota os resultados do vetor de bytes retornado.

GetRequest é usado por um processo servidor para obter requisições de serviços, quando o servidor tiver sido invocado a operação especificada, ele usa o *sendRply* para enviar a resposta ao cliente. Quando o cliente faz o recebimento da mensagem, a operação *doOperation* original é desbloqueada para que a execução do programa cliente continue.

As informações que serão transmitidas em uma mensagem de requisição ou em uma mensagem de resposta, apareceram na Figura a seguir:

messageType	*int (0=Request, 1=Reply)*
requestId	*int*
remoteReference	*RemoteRef*
OperationId	*int ou operação*
arguments	*// vetor de bytes*

Estrutura da mensagem de requisição-resposta.

O primeiro campo indica se o tipo de mensagem ou ***mensageType*** é um ***request ou Reply.*** O segundo campo, ***requestId***, contém um identificador de mensagem. Um ***doOperation*** no cliente gera um requestId para cada mensagem de requisição, o servidor irá identificar as respostas, copiar os identificadores das mensagens de resposta correspondentes. Isso permite que o ***doOperation*** verifique se há uma mensagem de resposta e se é o resultado da requisição atual e não de uma chamada anterior atrasada. No terceiro campo temos uma referência remota (RemoteReference). No quarto campo temos o identificador de operação (OperationId) a ser invocada. Citando um exemplo, as operações podem ser numeradas como 1, 2, 3…; se o cliente estiver usando uma linguagem comum que suporta reflexão, uma representação da operação em si, pode ser colocada neste campo.

Modelo de falhas de requisição-resposta:

- Se as três primitivas doOperation, getRequest e sendReply forem implementadas via datagramas UDP, elas sofrerão falhas de comunicação. Ou seja

- Sofrerão falhas por emissão.

- Não há garantia de entrega das mensagens na ordem de envios

Além disso, o protocolo sofre falhas nos processos, supomos que os processos têm falhas de colapso, quer dizer, podem parar e permanecer parados - e não irão produzir o comportamento contínuo.

Para levar em conta as ocasiões que o servidor falhou ou que uma mensagem de requisição foi perdida, o ***doOperation*** usa uma ação chamada (timeout) quando está esperando a resposta de um servidor. Essa ação será executada quando ocorre um tempo limite que depende das garantias de entrega oferecidas.

Descarte de mensagens de requisição duplicadas quando a requisição é retransmitida o servidor pode receber a primeira mensagem de requisição mais de uma vez. Citando um exemplo, podemos dizer que o servidor pode receber a mensagem, mas demorar mais do que o tempo limite do cliente para executar o comando e retornar a resposta. Isso pode levar o servidor a executar a operação mais de uma vez para a mesma requisição. Para evitar isso o protocolo é projetado de uma forma onde o mesmo possa reconhecer quando as mensagens

forem sucessivas (do mesmo cliente) e eliminar as requisições duplicadas. Se o servidor não tiver enviado a resposta ainda, não é necessário executar nenhuma ação especial - Ela transmitirá a resposta quando tiver terminado de executar a operação.

Sistemas peer-to-peer os sistemas peer-to-peer representam um paradigma para a construção de sistemas distribuídos em os dados e recursos computacionais são provinientes da colaboração de muitas máquinas na internet de maneira uniforme. Com o crescimento rápido da internet, aparecendo milhões de computadores e aumentando os acessos de usuários exigindo acesso a recursos compartilhados.

Um grande problema enfrentado pelos sistemas peer-to-peer é a distribuição de objetos de dados em muitos computadores e o subsequente acesso de maneira que seja possível equilibrar a carga de trabalho e tenha garantia de disponibilidade sem adicionar sobrecargas indevidas ao sistema.

Diversos middlewares para sistemas peer-to-peer têm surgido e disponibilizado capacidade para compartilhar recursos computacionais, armazenamentos de dados em computadores dentro dos “limites da internet” em uma escala global. Estão sendo exploradas novas técnicas de atribuição de nomes, roteamento, reaplicação dos dados e segurança existentes para que possa construir uma camada de compartilhamento de recursos confiável em um conjunto de computadores e redes inseguras e não confiáveis.

As plataformas desenvolvidas com o sistema peer-to-peer têm sido usadas para fornecer compartilhamento dos arquivos, uso cache Web, ajudando na distribuição de informações importantes dentro de empresas e outros serviços que exploram os recursos de milhares de máquinas ligadas à internet. Elas aparecem com sua maior eficácia quando usadas para armazenar conjuntos enormes de dados imutáveis. Seu projeto diminui sua eficácia quando feito para aplicações de dados mutáveis.

Os sistemas peer-to-peer se caracterizam da seguinte maneira:

- Seu projeto garante que cada usuário contribuir com recursos para o sistema

- Embora eles possam diferir nos recursos que contribuem, todos os nós em um sistema peer-to-peer têm as mesmas capacidades e responsabilidades funcionais.

- Seu funcionamento não depende da existência de quaisquer sistemas administrados de forma centralizada.

- Eles podem ser projetados de modo a oferecer um grau de anonimato para proteger os provedores e os usuários.

- Um problema importante para seu funcionamento eficiente é a escolha de um algoritmo para distribuição dos dados em muitas máquinas e o subsequente acesso a eles, de uma maneira que equilibre a carga de trabalho e garanta a disponibilidade sem adicionar sobrecargas indevidas

Middleware peer-to-peer Nesta terceira geração a característica de camada de middleware para o gerenciamento de recursos distribuídos em uma escala independente de aplicativos. Agora, várias equipes concluíram o desenvolvimento, avaliação e o refinamento das plataformas de middleware peer-to-peer e as demonstraram ou implementaram em diversos serviços de aplicativo. Essas plataformas são projetadas para alocar recursos (objetos de dados, arquivos) em um conjunto de computadores amplamente distribuídos em toda internet e para direcionar a eles em nome dos clientes, retirando dos mesmo as decisões sobre posicionamento de recursos e a necessidade de conter informações sobre o posicionamento de recursos e a necessidade de conter informações sobre o paradeiro dos recursos que exigem. Ao contrário dos sistemas de segunda geração, eles dão garantias dos envios de pedidos com um número limitado de passos intermediários de rede.