****

**PCS3818 - Engenharia de Sistema de Computação**

**Relatório da Atividade 6**

André Hideki Gashu Nishimura - 11325285

Gabriel Brandão de Carvalho - 11261941

Kevin Taiyo Onishi - 11261361

Marco Aurélio Condé Oliveira Prado - 11257605

Silas Lima e Silva - 11262233

William Abe Fukushima - 11261771

São Paulo

2023

1. **Introdução**

Através deste relatório, pretendemos refletir, discutir e consolidar os conhecimentos teóricos obtidos em aula relativos à avaliação da qualidade de um sistema ou software.

Para as análises propostas, selecionamos o atributo de qualidade **desempenho**. Discorreremos a respeito do que é dito sobre este atributo no livro “Software Architecture in Practice”, de Len Bass, Paul Clements e Rick Kazman, focando principalmente no que diz respeito à Lista de Verificação de Projeto proposta.

Ao final da atividade, esperamos ter compreendido e fixado devidamente os conceitos abordados durante a aula teórica.

1. **Categorias de decisões de projeto**

As categorias de decisões de projeto servem para organizar logicamente as preocupações do arquiteto sobre o sistema, salientando os aspectos mais problemáticos do projeto que devem receber maior atenção. Elas são divididas em: alocação de responsabilidades, modelo de coordenação, modelo de dados, gerenciamento de recursos, mapeamento entre elementos de arquitetura, decisões de tempo de ligação e escolha de tecnologia.

* 1. **Alocação de responsabilidades**

As decisões desta categoria identificam as responsabilidades importantes do sistema, como funções básicas, infra-estrutura arquitetural e atendimento de qualidade, e as alocam nos diferentes componentes. Para isso, se utiliza vários tipos de estratégias alinhadas ao tipo de sistema projetado, realizando, por exemplo, agrupamento de componentes seguindo funções semelhantes ou seguindo modos de operações de sistema, modelagens dos objetos reais, etc.

* 1. **Modelo de coordenação**

O modelo de coordenação indica os mecanismos pelos quais um elemento de software interage com outro. Sendo que suas decisões envolvem elementos do sistema que devem coordenar, ou são proibidos de coordenar, determinação das propriedades de coordenação e a escolha dos elementos de comunicação dessa coordenação, como se são stateful ou stateless, síncronos ou assíncronos, entre outros.

* 1. **Modelo de dados**

O modelo de dados refere-se a como estes são representados e interpretados no sistema. Neste modelo, tratamos de escolher as abstrações dos dados, junto com suas operações e abstrações, determinando como itens de dados são criados, inicializados, manipulados e etc. Além disso, o modelo é responsável por compilar os metadados necessários para interpretação consistente dos dados e a organização dos dados, como vai ser o armazenamento e o mapeamento desses dados.

* 1. **Gerenciamento de recursos**

Para decisões de gerência de recursos deve-se entender e balancear o uso dos recursos compartilhados entre os componentes do sistema. Como recursos entende-se desde aspectos de hardware até aspectos de software, por exemplo, uso da CPU, espaço de memória, acesso ao barramento, locks de sistema, buffers de software e thread pools.

 Essas decisões muitas vezes consistem em identificar quais são os recursos disponíveis e entender o limite de cada um, facilitando, em seguida, a determinação dos componentes do sistema que farão a gerência de cada recurso. Ao se aprofundar nas especificidades de cada recurso, deve-se definir como o compartilhamento será feito e como serão resolvidos os casos de disputa, aplicando um modelo de priorização, por exemplo.

* 1. **Mapeamento entre elementos de arquitetura**

O mapeamento na arquitetura de software se refere às relações na e conexões entre diferentes elementos arquitetônicos. Sendo que há dois tipos de mapeamentos que uma arquitetura deve fornecer uma delas sendo o mapeamento entre elementos em diferentes tipos de estruturas da arquitetura e o mapeamento entre elementos de software e elementos do ambiente.

 O primeiro tipo de mapeamento, por exemplo, pode mapear as unidades de desenvolvimento até as unidades de execução. Já o segundo tipo, pode mapear os processos para as CPUS e onde os mesmos serão executados.

* 1. **Decisões de tempo de ligação**

As decisões de tempo de ligação referem-se às decisões tomadas durante o processo de desenvolvimento de software que estabelecem faixas de variação permitidas. Essas decisões podem ser vinculadas em diferentes momentos do ciclo de vida do software, tais como tempo de projeto ou tempo de execução. A decisão de tempo vinculada determina o escopo, o ponto no ciclo de vida e o mecanismo para alcançar a variação.

 Além disso, cada uma das sete categorias de decisões de arquitetura tem uma decisão de tempo de ligação associada. Ao tomar decisões de tempo de ligação, é essencial considerar os custos de implementação da decisão e os custos de modificá-la após a implementação.

* 1. **Escolha de tecnologia**

A escolha da tecnologia é uma parte essencial do processo de design de arquitetura intencional. A seleção de tecnologia pode ser feita por outros, o que se tornará uma restrição à tomada de decisões em cada uma das sete categorias, ou pelo arquiteto, que deve escolher uma tecnologia adequada para realizar decisões em cada categoria.

O processo de escolha da tecnologia envolve várias considerações, incluindo a identificação das tecnologias disponíveis que podem cumprir os requisitos do projeto, avaliando se as ferramentas disponíveis são adequadas para apoiar o desenvolvimento, avaliando o nível de familiaridade interna e apoio externo à tecnologia, e analisando os efeitos colaterais da seleção de uma tecnologia, tais como modelos de coordenação necessários e oportunidades de gerenciamento de recursos.

Além disso, determinar a compatibilidade de uma nova tecnologia com a pilha de tecnologia existente é crucial. Isto inclui avaliar se a nova tecnologia pode operar ao lado ou em cima da pilha tecnológica existente, comunicar-se com ela e ser monitorada e gerenciada eficientemente.

1. **Descrição dos conceitos relacionados com desempenho**
	1. **Introdução**

Quando falamos de desempenho, uma das grandes palavras-chave envolvidas é “tempo”. Em grande maioria dos cenários relativos a qualquer sistema, é desejável que a tarefa a ser executada seja realizada no menor tempo possível, com o melhor resultado possível. Este conceito também está fortemente relacionado, por exemplo, com a noção de escalabilidade. Ao aumentarmos o volume de operações realizadas por um sistema, é de interesse de seus responsáveis que o seu desempenho seja afetado o mínimo possível.

 O desempenho começa a ser medido no momento em que um evento é disparado no sistema, onde a resposta ao evento requer o consumo de recursos como o tempo.

* 1. **Concorrência**

Concorrência ou paralelismo ocorre quando o sistema cria uma nova thread, pois cada thread é uma sequência de controle independente. Este conceito pode gerar suporte para atender múltiplos usuários simultaneamente. Para utilizar a concorrência, é necessário ter em mente seus conceitos durante a programação e na utilização de sistemas que possuem infraestrutura paralela. Concorrência aumenta drasticamente o processamento porque evita que atrasos se propaguem de um processo a outro, mas é necessário tomar cuidado com a identificação de condições de corrida ao utilizá-la.

* 1. **Controle da demanda de recursos**

Para controlar a demanda de recursos,podemos definir uma taxa que os recursos podem demandar do sistema ou limitar a taxa que o sistema atende a esses chamados. Entre as táticas estão

* **Gerenciar a taxa de amostra:** É possível reduzir a taxa que o sistema recebe dados, considerando o quanto seu sistema está disposto a perder a fidelidade com os dados reais, já que quanto maior a taxa de coleta, mais próximo do real.
* **Limitar a resposta a eventos:** Quando há eventos discretos com uma alta taxa de acontecimento, é possível haver filas no sistema, para resolver isso, você pode limitar a resposta do sistema à tais eventos, para isso você deve considerar, o quão danoso para o sistema é perder esse evento(adotando filas se necessários) e como tratar os eventos que não foram computados(se devem ser notificados, logados, etc.).
* **Priorizar eventos:** Caso haja diferença de importância entre eventos, é possível rankear estes de modo que o sistema dê prioridade aos mais urgentes, fazendo com que os eventos com menos prioridade sejam ignorados em caso de falta de recursos e melhorando o desempenho médio.
* **Reduzir o overhead:** Reduzir o número de passos intermediários em uma operação, pode ajudar a reduzir a latência do sistema, isso porque, com menos passos intermediários, o processamento necessário a ser feito pelo sistema diminui, já que teremos menos protocolos e menos cabeçalhos a serem decodificados pelo sistema durante o processamento.
* **Limitar os tempos de execução:** Você pode limitar o tempo de uso de recursos por um certo evento/algoritmo, para isso, deve-se considerar se o resultado chegado é bom o bastante no tempo que delimitamos para tal tarefa ser feita, não sendo uma matemática tão precisa.
* **Aumentar a eficiência de recursos:** Melhorar os algoritmos do sistema reduz a latência geral.
	1. **Gerenciamento de recursos**

 Outro grupo de técnicas empregado a fim de garantir um melhor desempenho do sistema é o gerenciamento de recursos. Administrar os recursos disponíveis de uma forma eficiente e balanceada é fundamental para garantir a lisura de qualquer processo, principalmente em cenários onde a demanda por estes não pode ser controlada.

 Há uma série de práticas que podem ser adotadas quando falamos de gerenciamento de recursos. Dentre eles, o livro-texto utilizado como referência cita:

* **Aumentar a quantidade/qualidade dos recursos disponíveis:** Dentre as abordagens citadas, é provavelmente a mais simples. Melhorar a quantidade de processadores e utilizar memórias mais rápidas, por exemplo, é uma ótima forma de se obter um ganho de desempenho em um sistema. Esta abordagem fica limitada, porém, à questões orçamentárias e de projeto.
* **Redundância de recursos computacionais:** Um dos principais objetivos ao utilizarmos a redundância de recursos computacionais é distribuir processamento. Desta forma, as cargas computacionais são balanceadas, o que resulta, por exemplo, em servidores menos sobrecarregados.
* **Redundância de dados:** São citadas duas formas de redundância de dados: cópia para cache e cópia para replicação. A cache são cópias de dados inseridos em componentes de memória com maior velocidade de acesso, de modo a acelerar os processos. Já a replicação de dados consiste em armazenar múltiplas cópias do mesmo dado em unidades de memória separadas com o objetivo de reduzir o número de disputas causadas por vários acessos simultâneos.
* **Controle do tamanho de filas:** Esta técnica controla o número máximo do tamanho das filas de chegadas de eventos que pode ser processada. A adoção desta técnica requer a implementação de uma política para lidar com quando as filas sofrem “overflow” e decidir se não responder a um evento perdido é aceitável para o sistema. Geralmente, a tática é utilizada juntamente da limitação de respostas a eventos.
* **Escalonamento:** Processos em um sistema competem por recursos, e definir a ordem em que estes serão distribuídos é fundamental para garantir um bom desempenho. Deve-se utilizar métodos para garantir que as prioridades de uso dos recursos sejam bem definidas, a fim de garantir que os processos sejam executados em tempo ótimo.
1. **Comentários sobre a Lista de Verificação de Projeto para desempenho**

Deve-se mapear todos os processos do sistema que possam impactar diretamente o desempenho do mesmo em maiores níveis. Dentre os principais para se listar, incluem-se processos altamente utilizados, com um grande carregamento de dados ou com necessidade de tempo crítico para resposta. Além disso, processos relacionados ao gerenciamento do controle de execução de threads devem ser analisados também, dado sua relação com os processos citados anteriormente.

A partir disso, pode-se criar mecanismos para coordenar esses processos da forma mais eficiente possível, priorizando tarefas com tempo crítico, sabendo quais podem ser paralelizadas, determinando parâmetros de comunicação como estados *(stateful/stateless)*, sincronização (síncrona ou assíncrona), latência, etc.

Os dados que foram identificados como frequentemente acessados podem ser remodelados para melhorar o desempenho do sistema, se feito da forma correta. Dentre as possíveis soluções, são citadas particionar os dados e reduzir os requisitos para processamento (criação, inicialização, manipulação e persistência) dos dados.

O mapeamento da arquitetura do sistema ajuda a determinar se os processos com grande necessidade de processamento estão designados aos processadores com maior capacidade. Além disso, saber onde e como aplicar concorrência e controle das threads pode influenciar.

Determinam-se os recursos críticos ao desempenho do sistema para que, desta forma, seja possível garantir melhores condições de monitoramento e gerenciamento dos mesmos impactando minimamente o sistema.

Por fim, é importante ressaltar que a escolha correta das tecnologias é um fator importante, pois é nessa etapa que se garante que você terá condições para aplicar todos os conceitos citados anteriormente.

1. **Comentários relevantes**

Através do desenvolvimento desta atividade, foi possível compreender de forma mais profunda como funcionam e influenciam os atributos de qualidade de sistema nas decisões de arquitetura e estruturação do projeto.

Além disso, através do livro, foi possível estudar um atributo de qualidade de forma mais detalhada, para nosso caso foi escolhido o atributo de desempenho. Através da análise do cenário, pudemos perceber como e quais os aspectos que influenciam o desempenho do projeto.

Falta a seção de Referências.