8º Cinfotec UNICAMP
26 e 27 de junho de 2019

SIGAD – Superando os Desafios de Performance de um Sistema Corporativo

Maisa Silva – maisasil@unicamp.br (SIARQ);
Eliel Goncalves – eliel@unicamp.br (DAC)

Introdução: o Sistema SIGAD é o gestor de documentos convencionais e digitais produzidos e/ou recebidos pela Unicamp, atendendo usuários (funcionários, alunos e professores) através de interface web, e outros sistemas através de biblioteca de serviços, o que faz da performance do sistema requisito prioritário, pois impacta diversas atividades da universidae. Nesse trabalho compartilhamos as principais ações realizadas nesse propósito, fornecendo um conjunto de pontos de decisões importantes.

Objetivo: a arquitetura atual do SIGAD é construída sobre a plataforma Java, utilizando definições JSF, EJB e JPA na sua implementação. A camada de interface é construída com os componentes de tela do framework IceFaces e o SGBD é o DB2. Esta configuração do sistema atende atualmente a 2.500 usuários e se integra com 2 sistemas, correspondendo a demanda de 35.000 documentos anualmente protocolados e 300.000 trâmites anuais.

Diante desse contexto a equipe SIARQ sempre se atentou a feedbacks relacionados a performance, os quais em geral eram relacionadas a uma funcionalidade específica do sistema, por exemplo: consulta de documentos pelo filtro nome de interessado. Dessa forma muitos problema foram superados com melhorias em banco de dados ou simplificação da lógica de negócio.

Resultado: a seguir são listados os principais relatos de baixo desempenho e as ações tomadas. Esse trabalho contou com apoio direto da equipe da DAC.

1 – Demora na consulta por nome de interessado ou assunto. O filtro dessas consultas vêm de campos livres, e para que os dados informados fossem comparáveis com os perisistidos eram aplicadas as funções lcase e translate. Para melhoria da performance retiramos a chamada a essas funções e passamos a persistir os dados já tratados em uma nova coluna.

2 – A tela inicial do sistema apresentava demora para ser carregada, isso se devia ao resumo das atividades pendentes do usuário presente nessa tela. O cálculo das pendências era feito em tempo real e em todo momento que a tela fosse acessada. Para melhoria da performance usamos Summary Tables que computam e armazenam as estatísticas de pendências periodicamente.

3 – Verificamos um volume muito grande nos logs de acesso da aplicação, ocasionado por inúmeros requests idênticos realizados na mesma fração de segundo. Após investigação verificamos que esse comportamento era provocado pelo framework Icefaces e que o efeito para o usuário era de um 'pico de lentidão'. Esse problema foi contornado ao desabilitar a propriedade 'com.icesoft.faces.synchronousUpdate'

4 – Outras ações realizadas foram: levantamento e exclusão de índices não utilizados, exclusão de caches não utilizados, criação de índices para consultas lentas e estruturação do ambiente de monitoramento Zorka e Zabbix.

Conclusão: Muitas ações tiveram impacto em funcionalidades específicas do sistema, sendo a ação de desabilitar a propriedade do IceFaces 'com.icesoft.faces.synchronousUpdate' a de maior impacto geral, já que antes o sistema era onerado por rajadas de requests sem propósito.

Atualmente o sistema opera muito bem e sem relatos de baixa performance. Mas, com a previsão de implantação de novas funcionalidades e o aumento de usuários, são estudadas ações como a migração da atual arquitetura monolítica para arquitetura distribuída. E principalmente a mudança da interface para abordagens stateless.