Enhanced Server Fault-Tolerance for Improved User Experience

André Esteves
nº3412

David Monteiro
nº3400

Introdução

- É proposto uma arquitectura de servidor Web dividida que tolera perfeitamente tanto falhas na Web proxy como no servidor back-end.
- A conexão TCP e sessões do cliente são preservadas, e a recuperação para servidores alternativos em caso de falhas do servidor é rápida e transparente para o cliente.
- A arquitectura oferece suporte para aplicações de servidor deterministicas e não-determinísticas.

Introdução

- As falhas da proxy ou do servidor é feita no ultimo servidor e transparente para o cliente;
- O failover é bastante rápido;
- Os estados e sessões dos clientes são guardados de modo a permitir uma recuperação rápida.

Background

Evolução

Consequencias:

- Utilizadores não necessitam de instalar, fazer a manutenção ou upgrade;
- Fácil acesso a aplicações em qualquer lado;
- Partilha de informação entre utilizadores separados geograficamente;
- Simplifica processo de enviar updates a cliente

TCP Splice

- Proposto para melhorar o desempenho das web proxies;
- Permite a proxy levar info entre o cliente e o servidor manipulando o cabeçalho do pacote;
- Torna a latência e computacional um pouco maiores que reencaminhamento IP;
- Não necessita de buffer na proxy

Estabelecer TCP Splice

- Cliente conecta á proxy, esta aceita a conexão e recebe pedido;
- Proxy faz autenticação e usa layer 7 routing para seleccionar o servidor e cria uma nova conexão TCP;
- Proxy envia o pedido do cliente ao servidor. As duas conexões são unidas;
- Após as duas conexões serem unidas o pacote vai do cliente para o servidor, com mudanças apropriadas no cabeçalho.

Desvantagens do TCP Splice

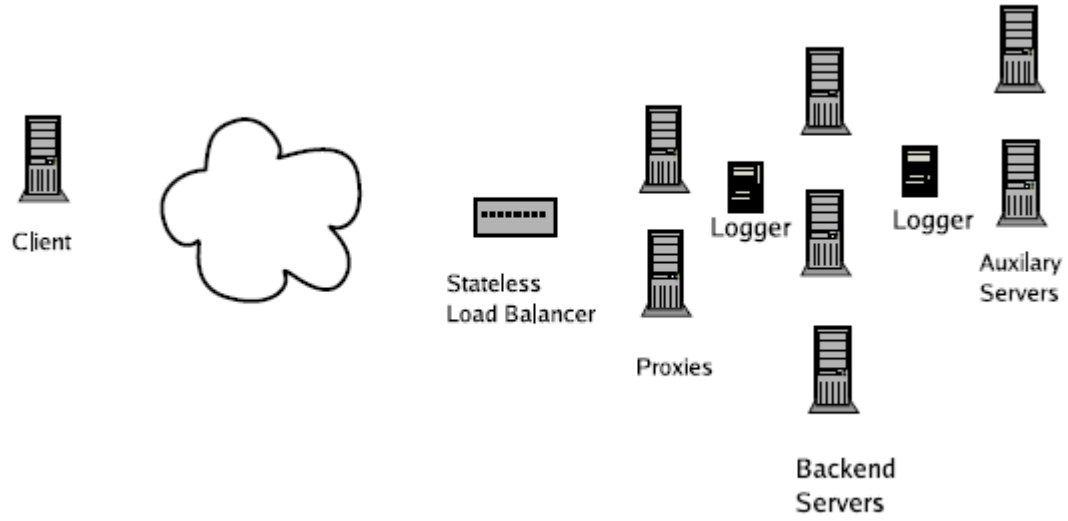
- O tráfego entre cliente e servidor leva a que a proxy seja escalável e a informação afunilada;
- Esta arquitectura não é tolerante a falhas. Se uma conexão TCP falha as restantes falham e o cliente tem de se conectar novamente.

Updates ao TCP Splice

- Replicated TCP Splice
  - Informação do estado é replicada em multiplas proxies.
  - Oferece escalabilidade e tolerancia a falhas.

- Split TCP Splice
  - Função splice é dividida em duas unioes unidireccionais.
  - Os pacotes com destino ao servidor vão pela proxy.
  - Os pacotes de resposta são unidos no servidor e enviados por uma proxy.

Arquitectura do sistema



Componentes de uma arquitectura Web Server

Arquitectura do sistema (cont.)

- Stateless load balancers: distribuem os pedidos dos clientes para as proxies.
- Proxies: realiza a camada de routing, união TCP, reunião durante a recuperação.
- Backend servers: processa pedidos clientes, envia as respostas de volta, envia o estado de sessão.
- Loggers: torna o log do tráfego transparente, detecta falhas nos backend server.
- Auxiliary servers: servidores adicionais com os quais os backend servers podem comunicar para ajudar nas respostas aos clientes.

Arquitectura do sistema (cont.)

- Os pedidos dos clientes são processam os pedidos dos clientes.
- Cada backend server pode processar um certo número de pedidos.
- Por vezes é necessário comunicar com outros servidores para responder aos pedidos.
- A cada transação de pedido/resposta é atribuido um ID.
- O ID é criado através do IP do cliente e do porto utilizado.

Arquitectura do sistema (cont.)

- Para facilitar a recuperação existem dois tipos de sistemas onde os pacotes IP estão logados.
- O primeiro é entre a proxy e o backend server, onde o cliente faz pedidos e este são respondidos pelo servidor onde está ligado.(front-end logger).
- O segundo é entre um backend server e um servidor auxiliar, onde este faz pedidos e é respondido pelo servidor onde está ligado.
- Os dois loggers aparecem na figura anterior que representa a localização destes.

Logging, Transactionalization and Tagging

- Todos os bytes destinados aos backend server vindos de um cliente passam por um front-end logger.
- O front-end logger usa aplicações especificas de configuração de ficheiros para saber onde começa e onde acaba um pedido.
- Cada par pedido/resposta do cliente é considerada uma transacção.
- O mapeamento entre as transações da camada de aplicação e a sequência de números TCP é necessário para migrar de uma conexão TCP para uma alteração de backend server caso ocorra uma falha.

Tags mais utilizadas

- **Determinista/não determinista:** uma transacção determinista é aquela que produz exactamente a mesma resposta e causa o mesmo efeito quando o backend server é reposto por um alternativo.
- **Leitura/Actualização:** Esta tag indica quando uma transacção muda de estado. Um pedido de leitura normalmente não tem qualquer efeito colateral, ao contrário um pedido para escrita pode mudar o estado do servidor.

Tags mais utilizadas (cont.)

- **Idempotente/não-idempotente:** uma transacção idempotente é aquela que produz o mesmo output e o mesmo efeito se for processada uma ou mais vezes.

Sincronização e Re-união

- O front-end logger grava os pedidos e as respostas para um backend server que monitoriza as falhas.
- A recuperação de falhas consiste em sincronizar o estado da conexão do cliente, a re-união dessa conexão e a sincronização do estado da aplicação do backend server alternativo.
- A sincronização do estado da conexão do cliente e o estado da aplicação é feita usando duas peças chave: o último ACK do cliente que foi guardado no logger e o último byte do servidor gravado no logger.

Suporte da aplicação

- É necessário suporte da aplicação para recuperar dos erros do servidor.
- É necessário que a aplicação transfira por cada transacção a informação desta transacção para o backend server auxiliar que está a correr a aplicação.
- Para melhor eficiência é preferivel que a aplicação mantenha apenas o estado da sessão.
- Outro requesito para a aplicação é que seja incluído um ID de transacção em cada pedido.

Não determinismo

- O não determinismo implica que cada vez que um pedido é feito tenha uma resposta diferente.
- O não determinismo pode ser um problema se o backend server falhar quando apenas tenha sido transmitida parcialmente a resposta.
- A não ser que haja alguma forma de gerar uma resposta idêntica não vai ser possível enviar o resto da resposta ao cliente.
- Nesta abordagem o problema do não determinismo foi tratado de duas formas.

Não determinismo (cont)

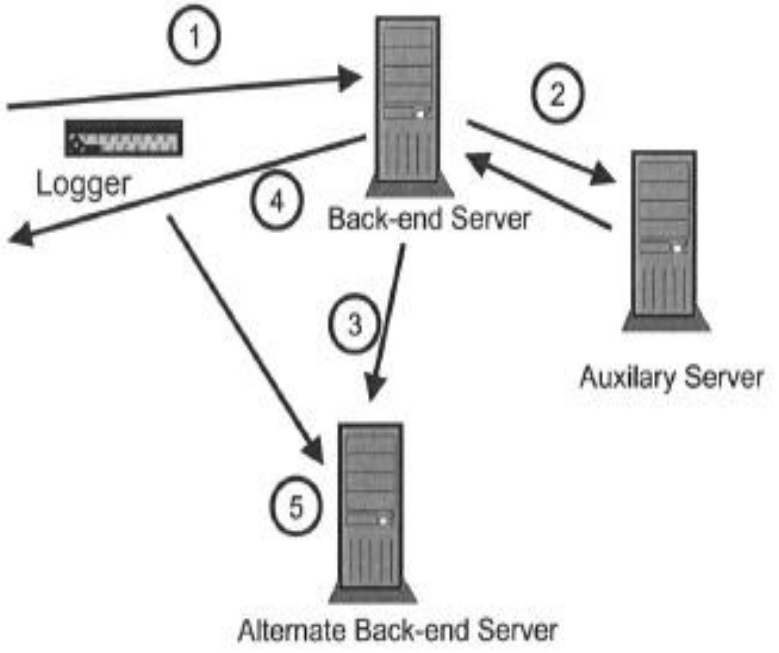
- Se é sabido à partida que uma determinada transacção é não-determinista a aplicação certifica-se que antes de enviar a resposta existe uma das seguintes verdades: (1) a totalidade da resposta está guardada num servidor backend alternativo, (2) apenas o estado da informação é copiado para o servidor alternativo e a partir deste é possível criar uma resposta determinista.

Detecção de falhas adaptativa

- Um detector de falhas de um servidor backend baseia-se num logger que é responsável por gravar os pedidos e respostas desse servidor.
- Quando um erro é encontrado normalmente é enviado um timeout.
- Este timeout é criado dinamicamente por um detector que observa todos os pedidos e respostas.
- Para cada pedido o detector mantem dois timeouts em funcionamento.
- O primeiro é o tempo entre a recepção total do pedido e o inicio do envio da resposta.
- O segundo é a diferença de tempo entre o início e o fim da resposta.

Handle Requests

1. O pedido do cliente é recebido pelo servidor
2. O servidor pede ajuda ao servidor auxiliar caso seja necessário e aguarda pela resposta
3. Se o pedido for para executar um update ele é efectuado também no servidor alternativo
4. O servidor responde ao pedido do cliente
5. O logger indica ao servidor de backup para guardar o estado do pedido em caso de falha.



Recuperação de Falhas

- O logger detecta que o servidor falha e envia a ultima transacção completa ao servidor alternativo;
- Proxy desliga-se do servidor que falhou e indica ás outras proxies para fazerem o mesmo;
- Servidor alternativo verifica a ultima transacção que tem guardada e o ultimo estado que foi aplicado e comunica essa informação ao logger.

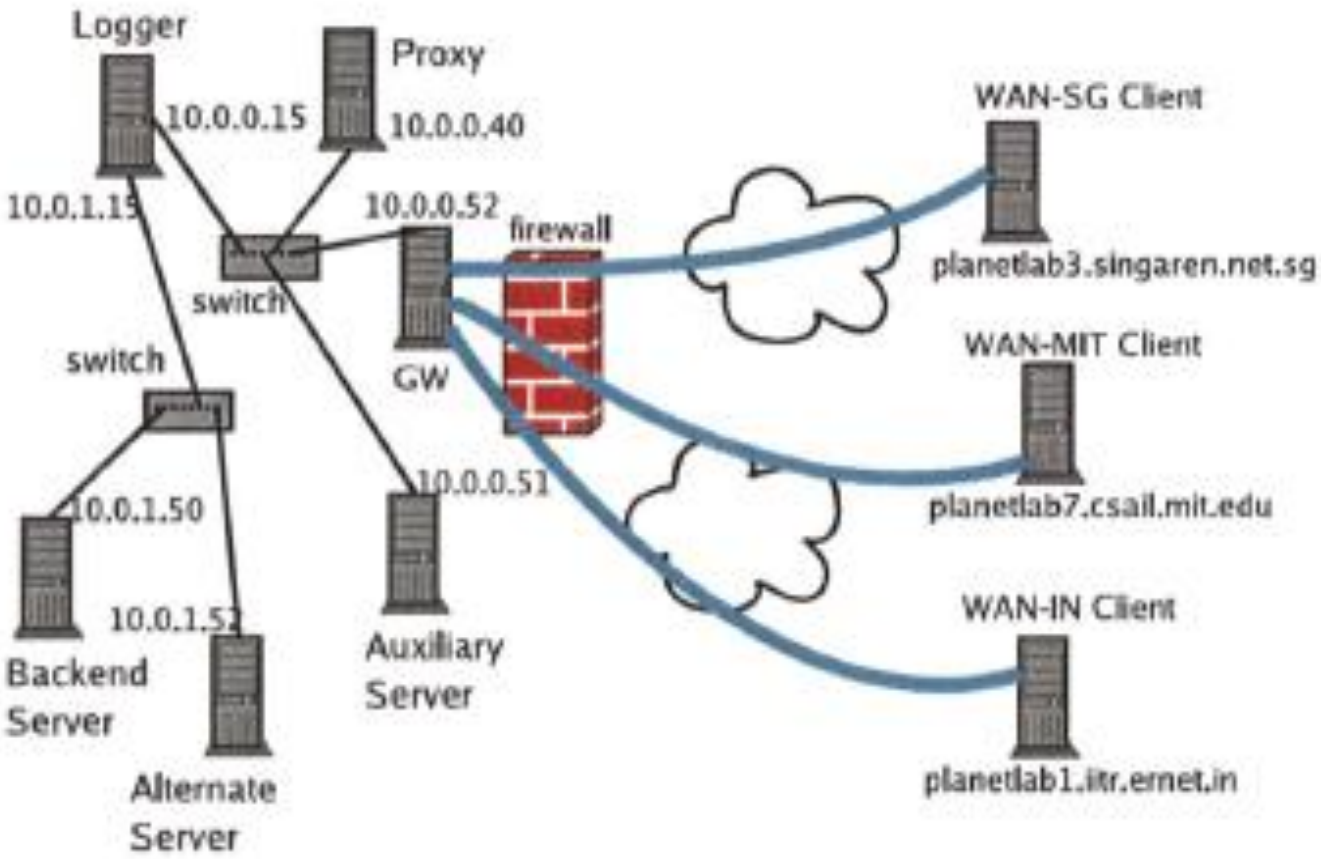
Recuperação de Falhas

- Sincronização com a conexão do cliente
- É guardado o ultimo ACK do cliente e é a partir dele que a re-transmissão de dados começa
- O logger já possui o pedido completo e verifica o que falta ser enviado
- É estabelecida uma união entre a ligação do cliente e a proxy para enviar os dados que faltam.

Recuperação de Falhas

- Sincronização com o servidor alternativo
- A informação dos estados pode não estar actualizada devido a lag na comunicação
- O servidor alternativo é actualizado até á ultima aplicação de estado de informação
- Transacções de leitura não são repetidas
- Isto é feito com base na informação do logger que está na proxy alternativa

Implementação



Resultados

	Average Time Taken (sec)						
	Traditional Architecture			Our Architecture			Overhead during Normal Operation
	No-Failure	With Failure	Failover	No-Failure	With Failure	Failover	
LAN	3.34	6.44	3.10	3.37	5.10	1.73	0.03 (0.89%)
WAN-MIT	23.0	35.9	12.9	23.6	25.10	1.50	0.6 (2.6%)
WAN-SG	68.1	105.5	37.4	68.5	71.45	2.95	0.4 (0.58%)
WAN-IN	299.2	373.5	74.3	297.7	314.7	17.0	(1.5)

Table 2: Time taken for performing one run of Action 1: Connecting to the login screen.

	Average Time Taken (sec)						
	Traditional Architecture			Our Architecture			Overhead during Normal Operation
	No-Failure	With Failure	Failover	No-Failure	With Failure	Failover	
LAN	3.72	6.69	2.97	3.83	4.77	0.94	0.11 (2.9%)
WAN-MIT	7.34	20.8	13.46	7.50	8.95	1.45	0.16 (2.17%)
WAN-SG	15.74	45.8	30.06	16.9	20.41	3.51	1.16 (7.3%)
WAN-IN	70.84	128.78	57.94	73.91	72.78	1.13	3.07 (4.3%)

Table 3: Time taken for performing one run of Action 2: Logging in; drafting and sending an email; and, logging out.

Fim