Um estudo empírico de composições de serviços:

efeitos da variação de redundância e confiabilidade dos papéis

MAC-0499 - Trabalho de conclusão de curso
Bacharelado em ciência da computação
Instituto de matemática e estatística - Universidade de São Paulo
Dezembro de 2016

Aluno: Gabriel Torres Gomes Pato
Orientador: Daniel Macêdo Batista

1 Resumo

O paradigma da computação orientada a serviços permite o desenvolvimento de aplicações modulares e distribuídas, propriedades desejadas no contexto atual onde softwares independentes de plataforma são desejados. Uma das formas de construir tais softwares é por meio da composição de serviços já existentes e disponíveis na web. O presente trabalho avaliou o efeito de diferentes estratégias de composição de serviços levando em conta requisitos não funcionais como tempo de resposta e taxa de sucesso.

2 Objetivo

O objetivo do presente trabalho é estudar na literatura o estado-da-arte na área de taxonomia de serviços web usando requisitos não funcionais (também chamados requisitos de qualidade de serviço, ou QoS) e realizar um estudo empírico de composições de serviços que atendam aos requisitos de tempo de resposta máximo e taxa de sucesso definidos pelo usuário.

3 Motivação

Acreditamos que a computação orientada a serviços, por ser um paradigma importante dentro da computação, em particular por permitir o desenvolvimento de aplicações distribuídas e interoperávies de forma rápida e com custo reduzido em ambientes heterogêneos [7], seja um objeto de estudos importante e de relevância cada vez maior num mundo onde a tendência é construir componentes de software independentes de plataforma a partir de serviços disponibilizados na web [10].

4 Conceitos Básicos

4.1 Orientação a serviços

Orientação a serviços, ou arquitetura orientada a serviços (SOA), é um paradigma de design de software no qual uma aplicação é composta de serviços. Os princípios de tal paradigma prezam pela separação de conceitos, ou separação de preocupações (do inglês, *separation of concerns*), ou seja, cada unidade do software criada dentro deste paradigma deveria resolver uma determinada tarefa, gerando um software final modularizado e fracamente acoplado. A cada unidade funcional deste tipo de software, dá-se o nome de **serviço**.

Dentro deste paradigma, os componentes de uma aplicação, ou serviços, provêem sua funcionalidade a outros componentes usando um protocolo comum de comunicação. Em grande parte das vezes, essa comunicação é feita através de uma rede.

Um outro aspecto relevante do paradigma é a ênfase em interoperabilidade e independência de plataforma, isto é, a funcionalidade de um serviço deve ser acessível a outros serviços escritos em linguagens diferentes ou rodando em sistemas operacionais diferentes, por exemplo.

Grandes sites da web como Facebook [11] e Twitter [12] disponibilizam serviços para que programadores possam por exemplo fazer buscas nos conteúdos publicados nesses sites utilizando as linguagens de programação de sua preferência. Com serviços bem definidos, uma aplicação que acesse tais sites continua funcionando mesmo que aspectos da interface gráfica desses sites, ou tecnologias de banco de dados, sejam modificados, o que é uma grande vantagem em relação a outras formas de programação envolvendo conteúdos disponibilizados na Internet.

4.2 Serviço web

Um serviço é definido pela W3C (World Wide Web Consortium) como um "*(...) software desenhado para oferecer interação entre máquinas, de forma interoperável, numa rede. O serviço web possui uma interface descrita num formato processável por máquina (WSDL – Web Services Description Language). Outros sistemas podem interagir com o serviço web da forma especificada na sua descrição usando mensagens SOAP, normalmente transportadas usando HTTP e utilizando serializações de arquivos XML e outros padrões relacionados à Web*" [1]. Um serviço web pode também ser definido, de forma simplificada, como qualquer software que está disponível na web e utiliza um sistema padronizado de mensagens em XML [2]. Outra forma de criar serviços e interagir com os mesmos é por meio do estilo arquitetural REST [13]. De forma similar ao SOAP, mensagens são transportadas usando HTTP porém a serialização pode ser feita em formatos diferentes do XML, como JSON por exemplo.

As características básicas de um serviço web são:

- está disponível numa rede (internet ou alguma intranet);
- utiliza um sistema padronizado de mensagens, como XML;
- não está atrelado a nenhuma linguagem de programação específica ou sistema operacional;
- é auto-descritível através de uma gramática (como WSDL);
- pode ser 'descoberto' através de um mecanismo de busca simples.

Para seu funcionamento, um serviço web se vale de alguns padrões abertos como HTTP, XML, WSDL e SOAP.

No caso de um sistema baseado em SOAP, os dados trocados nas mensagens são rotulados com XML, o protocolo SOAP transfere as mensagens e o WSDL descreve a disponibilidade do serviço. O protocolo HTTP transporta as mensagens pela rede.

A arquitetura dos serviços web possui alguns tipos de agentes, a saber: provedor de serviço, solicitante do serviço e registrador de serviços. O provedor de serviço é quem implementa o serviço e o disponibiliza na rede; o solicitante do serviço é quem vai utilizar o serviço prestado, estabelecendo uma conexão com o provedor e enviando uma requisição XML; o registrador de serviços é um ponto centralizador onde os provedores registram seus serviços e solicitantes procuram por serviços de seu interesse.

Os provedores de serviço disponibilizam seus serviços web na rede e os chamados *service brokers* podem ser usados para auxiliar os solicitantes a encontrá-los, utilizando o protocolo UDDI (*Universal Description, Discovery and Integration*) que oferece mecanismos para registro e busca de serviços web.

4.3 Composição de serviços

Uma vez que o paradigma SOA dita que os serviços sejam unidades de funcionalidade independentes, uma aplicação desenvolvida dentro deste paradigma pode ser criada a partir de serviços pré existentes fornecidos por terceiros, apenas unindo tais serviços de forma coordenada [6]. Este agregado de serviços é chamado **composição de serviços** que é, por sua vez, também um serviço.

As composições de serviço, em particular de serviços web, podem ser criadas manualmente por um desenvolvedor que analisa as propriedades de cada serviço e os requisitos da composição a ser criada. As entradas e saídas de cada um dos serviços são interligadas de forma coordenada para obter uma composição final que atenda aos requisitos iniciais. Além disso, uma composição de serviços pode ser criada de forma automatizada se valendo dos padrões existentes para a descrição e busca de serviços. Depois de especificados, os requisitos da composição são utilizados para selecionar os serviços disponíveis na web. Para tal, diversas abordagens são possíveis, como descrito em [7].

Do ponto de vista de um serviço único, o fato dele compor uma aplicação, ou seja, o fato dele fazer parte de uma composição de serviços, é transparente, já que ele responderá a uma requisição da mesma forma que ele responderia se um usuário solicitasse apenas este serviço. A comunicação entre serviços também ocorre de forma transparente para um serviço específico, já que o HTTP continua sendo usado para transportar as mensagens. As composições são criadas por programadores utilizando qualquer linguagem de programação com suporte a serviços.

4.4 Requisitos não funcionais

O conceito de requisito não funcional não possui uma definição única e objetiva e, no entanto, é utilizado há mais de 30 anos. Para tornar mais intuitiva a compreensão do que seriam requisitos não funcionais, começaremos definindo requisitos funcionais de um sistema.

Um requisito funcional se refere às funções que um sistema deve desempenhar ou "o que o sistema deve fazer"[3]. Uma segunda abordagem para a definição dá enfoque ao comportamento do sistema: requisitos comportamentais de um

sistema são "aqueles que especificam os *inputs* ao sistema, os *outputs* do sistema bem como relações comportamentais entre ambos, também chamadas de requisitos funcionais ou operacionais."[4] A partir destas definições, podemos tentar definir o que seriam requisitos não funcionais.

Um requisito não funcional seria algo que o sistema deveria conseguir fazer mas que não se refere às entradas e saídas dele especificamente. Características como tempo de resposta, segurança, confiabilidade, auditabilidade, custo, usabilidade, portabilidade, etc, são usualmente consideradas requisitos não funcionais. Nota-se que tais características impõem restrições muitas vezes subjetivas. Como quantificar quão seguro é um sistema? Ou quão confiável ou auditável?

No escopo deste trabalho adotaremos a definição de requisitos não funcionais a seguir: "requisitos que não estão particularmente focados na funcionalidade do sistema. Eles impõem limites no produto sendo desenvolvido e no processo de desenvolvimento e especificam restrições externas as quais o produto deve obedecer" (tradução livre de [5]).

No contexto dos serviços web, requisitos não funcionais são também chamados de qualidade de serviço ou QoS (do inglês *quality of service*) [8, 9].

Com base nos conceitos apresentados, é possível observar que, ao definir uma composição de serviços, quando há várias opções de serviços disponíveis e o objetivo a ser alcançado é cumprir requisitos não funcionais definidos por um usuário, a escolha pelo conjunto de serviços que vai ser usado não é trivial. Simplesmente cumprir as entradas e saídas do sistema não é suficiente. Passa a ser necessário avaliar se os serviços escolhidos conseguem criar um sistema que, por inteiro, atenda aos requisitos do usuário, preferencialmente sem realizar uma escolha gulosa que criaria todas as possíveis composições e pararia quando uma delas cumprisse os requisitos.

5 Experimentos preliminares

5.1 Motivação

Com o intuito de avaliar a relevância da criação de um catálogo de serviços web, contendo informações sobre desempenho, confiabilidade, agilidade na resposta, tipo de resposta e etc, realizamos uma coleta de dados durante requisições a um conjunto de serviços web e analisamos estes dados. Foram medidos tempo de resposta e percentual de sucesso nas requisições enviadas aos serviços. Este catálogo funcionaria como um banco de dados disponível publicamente para que usuários pudessem selecionar os serviços que melhor se adequam às suas necessidades. Atualmente, existem propostas de catálogos mas, na prática, são pouco utilizadas.

Assim, tais experimentos preliminares permitiram a avaliação de diferentes métricas de serviços web que auxiliaram nos experimentos que avaliaram os efeitos de variações de requisitos não funcionais na qualidade de composições de serviços. Uma motivação extra dos experimentos preliminares foi compreender melhor o processo de desenvolvimento orientado a serviços.

5.2 Materiais e métodos

Foram criados 3 serviços web, que foram hospedados no servidor Godzilla, localizado dentro da rede do IME. O servidor possui 16 GB de RAM, processador Intel® Core™ i7-2700K CPU @ 3.50GHz com 8 núcleos, 8 MB de cache, 2 discos SATA de 1 TB cada e 1 disco SSD de 120 GB. O sistema operacional instalado é o Debian GNU/Linux 6.0.10. Os serviços estão descritos abaixo:

- `Loop`: um serviço que apenas executa um loop vazio por uma quantidade grande de vezes. A motivação foi ter um serviço que tenha um tempo de processamento elevado por parte do servidor;
- `Soma`: um serviço que recebe dois inteiros, soma e devolve este resultado. Além desta computação simples, o serviço gera um número pseudo aleatório que decide se ele vai retornar um erro (resposta da requisição http com código 500) ou o resultado da operação (resposta da requisição http com código 200). A motivação foi ter um serviço que falha eventualmente;
- `Temperatura`: este serviço solicita a 3 outros serviços, disponíveis na internet, a temperatura na cidade de São Paulo e retorna seus valores. A idéia aqui foi avaliar o comportamento de um serviço que não depende apenas do servidor onde está hospedado, mas também da obtenção de resultados externos, que podem demorar ou falhar naturalmente. Os serviços utilizados são disponibilizados pelos sites `http://www.webservicex.net/globalweather.asmx`, `http://api.openweathermap.org` e `http://servicos.cptec.inpe.br`.

Foi escrito um código na linguagem PHP que invocava 1000 requisições a cada um dos serviços implementados. Ao receber a resposta de uma requisição, o código esperava 30 segundos antes de enviar a próxima. Essas requisições foram realizadas em um laptop Dell equipado com 8 GB de RAM, processador Intel® Core™ i5-5200U CPU @ 2.20GHz com 4 núcleos e 980 GB de disco, conectado à rede por cabo Ethernet (100Mbps). A conexão à Internet por sua vez era realizada por meio de um serviço residencial com taxa de 25Mbps. O sistema operacional instalado é o Ubuntu 14.04 LTS.

Os dados coletados foram: momento de envio da requisição, tempo de resposta em milissegundos, código da resposta http e mensagem recebida pelo serviço.

5.3 Resultados

As requisições aos serviços `loop` e `temperatura` tiveram 100% de sucesso. O serviço `soma` teve 48,9% de requisições bem sucedidas e 51,1% de requisições que falharam. Estas falhas foram todas decorrentes da implementação do serviço, feito para falhar em aproximadamente 50% das vezes.

Os tempos de resposta das requisições estão exibidos nas figuras 1, 2 e 3 para os serviços `loop`, `soma` e `temperatura`, respectivamente. No eixo horizontal

temos as classes de valores, em milissegundos, e no eixo vertical a quantidade de requisições cujo tempo de resposta ficou dentro daquela classe de valores (em unidades). Nota-se a pouca dispersão de valores para todos os serviços implementados.

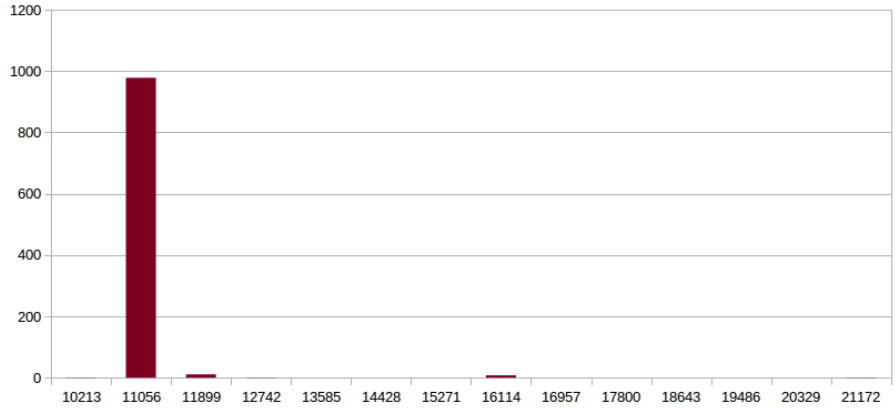

Figura 1: Histograma de tempos de resposta das requisições feitas para o serviço `loop`. Eixo horizontal em milissegundos, eixo vertical indica quantidade de requisições.

O serviço `temperatura`, o único composto por diversos outros serviços, foi o que teve maior tempo médio de resposta, 16593 milissegundos, enquanto o serviço `loop`, que realizava uma computação demorada, teve o segundo maior tempo médio de resposta, 10630 milissegundos, e o serviço `soma` foi o mais rápido, em média, levando aproximadamente 12 milissegundos.

Nas requisições ao serviço `temperatura` observamos que uma das fontes de informação, ou seja, um dos 3 serviços que compunham o nosso serviço, se mostrou indisponível durante toda a coleta. O serviço `temperatura` só conseguiu coletar parte da informação pretendida. Além disso, cada fonte de informação trazia os dados num formato diferente, não só em termos de organização, mas em termos de unidade de medida (Celsius, Fahrenheit e Kelvin). Estas temperaturas, ainda que fossem convertidades para uma mesma unidade, não apresentavam uniformidade de valores, ou seja, cada fonte fornecia uma temperatura diferente para a mesma cidade, São Paulo.

5.4 Discussão

Avaliando os dados obtidos, podemos notar a enorme variação no tempo de resposta para serviços disponíveis numa mesma máquina, dentro da mesma rede (médias de 12ms, 10630ms e 16593ms para `soma`, `loop` e `temperatura`, respectivamente). O tempo médio do serviço `temperatura` foi mais de 1300 vezes maior do que o do serviço `soma`, por exemplo. A variação era esperada por algumas razões: o serviço `temperatura` deve fazer 3 requisições a outros serviços antes

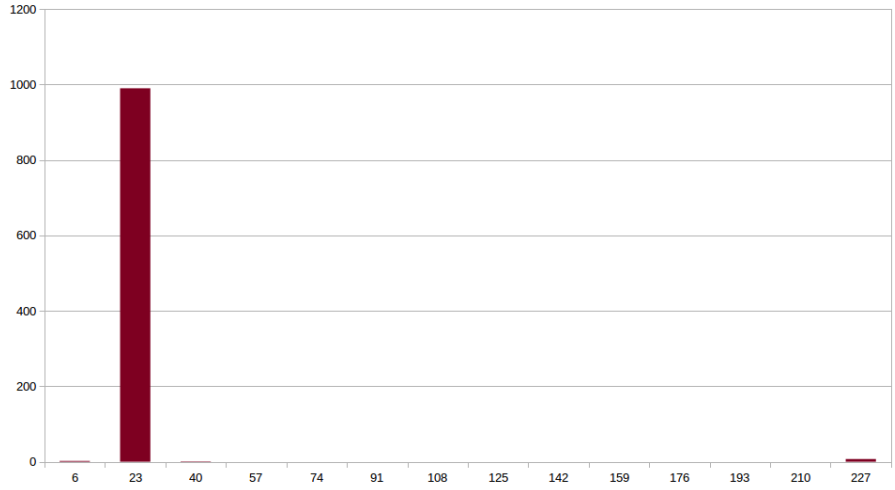

Figura 2: Histograma de tempos de resposta das requisições feitas para o serviço `soma`. Eixo horizontal em milissegundos, eixo vertical indica quantidade de requisições.

de completar sua execução e poder enviar uma resposta ao solicitante; duas destas requisições são feitas a servidores hospedados em outro país (EUA), o que tende a ter um maior tempo de resposta; estes serviços estão disponíveis publicamente, aumentando as chances de ocorrer uma grande quantidade de requisições simultâneas a seus servidores, aumentando também o tempo de resposta; o serviço `soma` realiza apenas um cálculo trivial entre 2 inteiros antes de responder à requisição.

Um outro aspecto avaliado foi o da confiabilidade dos serviços. O serviço `temperatura` se assemelha a um serviço composto real, onde a informação que ele disponibiliza depende completamente de outros serviços, muitas vezes sob controle de outras organizações e sujeitos a variação na disponibilidade. Supondo que o serviço retornasse a temperatura média fornecida pelos seus serviços componentes, nosso resultado teria um viés, uma vez que 1/3 da informação que seria utilizada para obter tal média esteve indisponível durante todo o tempo da coleta de dados. Chamamos este tipo de resultado do serviço de 'falha parcial' pois a computação do valor não foi completa. Já o serviço `soma`, conforme projetado, apresentou cerca de 50% de sucesso.

Todas estas informações nos mostram a relevância de se construir um catálogo de serviços, com descrições e meta informações a respeito destes. Acreditamos que apenas as informações apresentadas num documento em formato WSDL não fornecem informações suficientes para que usuários selecionem serviços que atendem às suas demandas. Não é do nosso conhecimento que haja um catálogo largamente utilizado e mantido na Internet, portanto, é importante que desenvolvedores que planejem utilizar serviços web em suas organizações tenham como

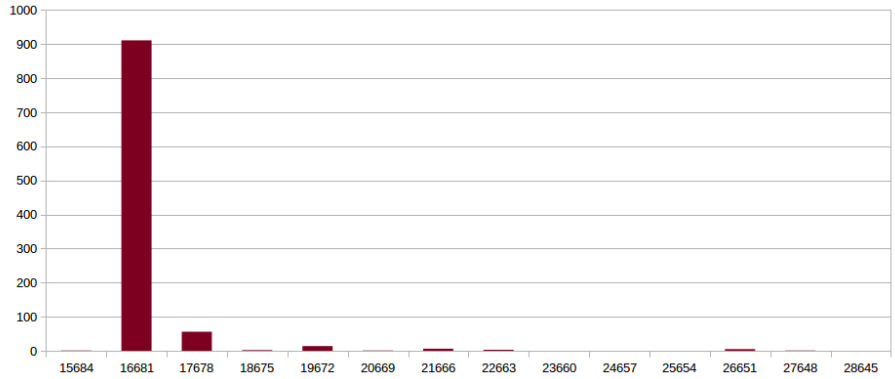

Figura 3: Histograma de tempos de resposta das requisições feitas para o serviço `temperatura`. Eixo horizontal em milissegundos, eixo vertical indica quantidade de requisições.

primeiro passo o projeto e desenvolvimento de um catálogo a fim de facilitar a busca por informações e de evitar a escrita de serviços redundantes.

6 Experimentos com composições

Esta seção apresenta os experimentos que foram realizados com diversos serviços e composições de serviços a fim de avaliar os efeitos de diferentes configurações relacionadas com redundância, com confiabilidade e com atraso dos serviços. Inicialmente são apresentados os serviços que foram implementados. Posteriormente são apresentadas as composições que foram construídas usando tais serviços. Em seguida é apresentado o procedimento que foi executado para fazer a coleta dos dados durante as execuções das composições. Por fim os resultados obtidos são discutidos. O hardware utilizado nestes experimentos foi o mesmo utilizado nos experimentos preliminares e que foi apresentado na Subseção 4.2 com a diferença de que agora o laptop conectava-se à rede via WiFi a 54Mbps. Também assim como nos experimentos preliminares, todos os códigos foram escritos nas linguagens PHP (serviços) e Python (executor das requisições).

6.1 Materiais e métodos

6.1.1 Serviços

Foram idealizadas e implementadas 6 classes de serviços que executam operações simples com números inteiros, a saber:

1. `Soma`: recebe dois inteiros, a e b, e devolve a soma deles, (a+b);
2. `Mod2`: recebe um inteiro n e devolve o resto da divisão de n por 2;

3. `Div`: recebe dois inteiros, a e b, e devolve o maior inteiro menor que (a / b), caso b diferente de 0, ou 0 caso contrário;

4. `Mult`: recebe dois inteiros, a e b, e devolve sua multiplicação (a * b);

5. `Oposto`: recebe um inteiro a e devolve seu oposto -a;

6. `Random`: não recebe parâmetros e devolve um inteiro entre 1 e 100.

Dentro de cada uma das classes, 16 serviços foram criados e nomeados sequencialmente (`soma_0`, `soma_1`, `soma_2`, `mod2_0`, `mod2_1`, etc...). Cada um destes serviços possuía valores próprios de confiabilidade, isto é, frequência com a qual uma requisição a ele era bem sucedida (status http 200), e atraso na resposta, isto é, tempo gasto para que a resposta da requisição fosse enviada, fosse ela bem sucedida ou não.

Para simular a confiabilidade dos serviços utilizamos um número pseudo-aleatório entre 1 e 100 obtido com a função *rand*(a,b) da linguagem PHP. Este número era comparado à confiabilidade, em porcentagem, estipulada para aquele serviço e, caso fosse maior do que ela, o serviço retornava erro (status http 500), caso contrario, o serviço retornava o valor computado e status http 200.

A simulação do atraso foi feita através de um laço vazio. Cada serviço executava um laço vazio antes de avaliar o resultado do teste da confiabilidade. O valor do atraso estipulado para o serviço era o número de vezes que o laço vazio executava. Para os fins deste trabalho, o atraso real em milissegundos não foi relevante, uma vez que queríamos apenas serviços com diferentes tempos de resposta.

Os valores de confiabilidade utilizados nos experientos foram 100%, 75%, 50% ou 25% para cada serviço. Os valores de atraso foram 1000000000, 10000000, 100000 ou 1000.

O número de cada serviço (O sufixo após o '_') determinava quais os valores de confiabilidade e atraso daquele serviço. A relação se encontra na Tabela 1. Essas informações foram mantidas em um catálogo simples que foi construído para os experimentos.

6.1.2 Composições

Foram criados 3 grupos de composições de serviços. Cada grupo era caracterizado pela quantidade total de papéis e pelo encadeamento das ações executadas pelos papéis. Um papel podia ser realizado por vários serviços. No nosso ambiente de experimentação, cada papel tinha 16 opções de serviços. Os parâmetros que variam dentre os membros de um mesmo grupo de composições eram apenas os parâmetros numéricos como confiabilidade, redundância e atraso máximo.

Cada composição foi descrita num arquivo XML composto por uma *tag* inicial, <total_papeis>, responsável por indicar quantos papéis faziam parte daquela composição, e um conjunto de 6 *tags*, encapsuladas por uma *tag* <papel>, que representava cada papel da composição (<id>, <acao>, <argumentos>, <delay_max>, <confiabilidade>e <redundancia>). Um exemplo de com-

Tabela 1: Numeração adotada para identificar os serviços dentro de uma mesma classe e seus respectivos valores de confiabilidade e atraso

Número do serviço	Confiabilidade	Atraso
0	100%	1000000000
1	100%	10000000
2	100%	100000
3	100%	1000
4	75%	1000000000
5	75%	10000000
6	75%	100000
7	75%	1000
8	50%	1000000000
9	50%	10000000
10	50%	100000
11	50%	1000
12	25%	1000000000
13	25%	10000000
14	25%	100000
15	25%	1000

posição está ilustrado na Figura 4. Nesse exemplo está sendo definida uma composição com 2 serviços que serão executados em sequência (o segundo serviço é executado apenas depois do primeiro terminar). O primeiro papel exige a execução do serviço `random`. O segundo papel exige a execução do serviço `oposto`. Os parâmetros de atraso máximo aceitável, confiabilidade mínima e redundância são respectivamente 747242905, 63 e 0 para o primeiro papel e 767976590, 25 e 0 para o segundo papel. Além disso, o segundo papel também recebia um argumento utilizado pelo serviço que iria realizá-lo.

A tag <acao> indicava qual das classes de serviço deveria ser utilizada para preencher aquele papel. Dentre as 16 possibilidades daquele grupo de serviços, um serviço era escolhido para preencher aquele papel, usando como critério a confiabilidade e o atraso máximo estipulados no papel respectivamente nas tags <confiabilidade> e <delay_max>.

A tag <redundancia> indicava quantas requisições, além da primeira, seriam feitas a serviços do grupo desejado. Se um determinado papel indicava que sua ação era `oposto` e possuía redundância 2, seriam executadas 3 requisições a serviços do grupo `oposto`, mas apenas 1 resultado seria utilizado. As requisições extras eram feitas em paralelo, com o uso de threads. Em casos como este, onde a redundância era maior que zero, o primeiro resultado válido era utilizado.

Caso nenhuma requisição tivesse sucesso, aquele papel era considerado como falho. Se papéis subsequentes dependessem deste para finalizar a composição, esta era interrompida e tida como falha.

Alguns serviços, como os das classes `soma` e `mult`, necessitavam de argumen-

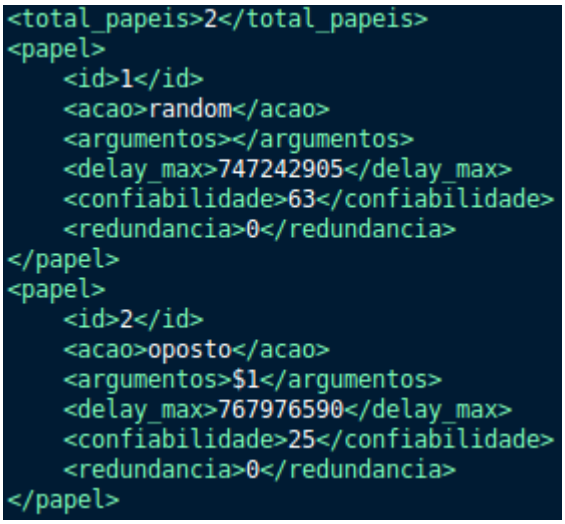

Figura 4: Exemplo de composição utilizando o XML adotado

tos para realizar as operações. Estes argumentos poderiam ser valores inteiros ou valores de retorno de algum dos serviços anteriores. Os inteiros eram declarados explicitamente dentro da tag <argumentos>. Quando o valor de retorno de algum papel era necessário, a referência era feita utilizando o caractere '$', seguido do <id>do papel ao qual se queria referir. Se o papel de <id>= 6 fosse utilizar o valor de retorno do papel de <id>= 3, na lista de argumentos do papel de <id>= 6 constaria o argumento '$3', por exemplo.

Composições de serviços também costumam ser apresentadas como fluxos de trabalho (*workflows*) ou grafos. A Figura 5 utiliza essa abordagem para apresentar a composição denonimada '1A'. Nesse formato, um vértice representa um papel ou o resultado final produzido pela composição, e um arco $a \rightarrow b$ implica que o serviço que realiza o papel `b` só pode ser requisitado após a saída do serviço que realiza o papel `a` ser recebida.

O grupo de composições 1 foi construído com 6 papéis, cada um com uma ação distinta, ou seja, todas as classes de serviços foram utilizadas na construção de cada composição do grupo. Todos os papéis possuíam valor 0 de redundância. A primeira composição deste grupo (composição 1A, Figura 2) foi criada e seus dados analisados. A partir dos resultados desta primeira composição, as demais composições do grupo foram criadas. Os detalhes são discutidos adiante, na seção que trata dos resultados.

O grupo 2 foi composto também de 6 papéis, todos com a mesma ação (`random`) e mesmos valores para confiabilidade e atraso máximo.

O grupo 3 foi composto de apenas 1 papel (`random`) com os mesmos valores de confiabilidade e atraso máximo dos papéis do grupo 2. A redundância das composições deste grupo variou de 5 a zero. A intenção foi comparar os resultados das requisições feitas em série (grupo 2) e em paralelo (grupo 3).

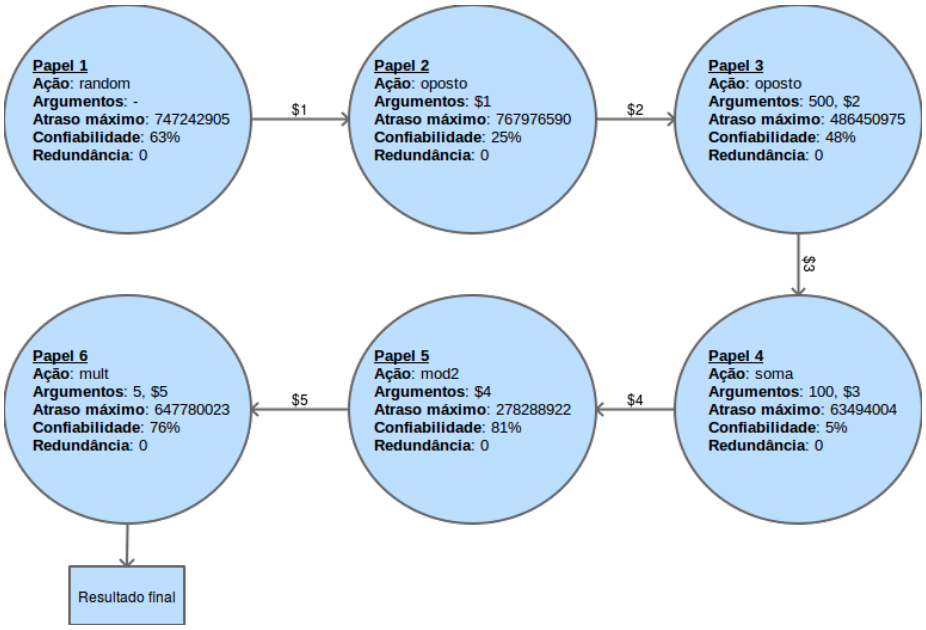

Figura 5: Exemplo de composição como um *workflow*

Os valores de confiabilidade e atraso máximo atribuídos a cada um dos papéis de cada uma das composições foram obtidos de forma pseudo-aleatória utilizando a função *randint*(a,b) da linguagem Python, com a = 1 e b = 100 para confiabilidade e a = 100 e b = 1000000000 para atraso máximo (note que existia a possibilidade de criarmos composições inviáveis caso atraso máximo >= 100 e <1000, uma vez que não havia serviços com atraso máximo < 1000).

A relação de todas as composições criadas está no Anexo 1.

6.2 Coleta de dados

A fim de realizar a coleta de dados foi desenvolvido um código que executava o seguinte algoritmo em 4 etapas:

1. Parsing do XML: um arquivo XML, descrevendo uma composição, era passado como entrada para o algoritmo, que extraía as informações necessárias para realizar a composição;

2. Atribuição dos serviços: para cada papel extraído do XML, um ou mais serviços eram selecionados, levando em conta a ação, a confiabilidade necessária e o atraso máximo permitido.

3. Execução dos serviços selecionados: nesta etapa, cada serviço escolhido para um papel era executado, o resultado desta execução era avaliado e os dados eram salvos.

4. Laço: utilizando a estrutura de dados obtida após a etapa de parsing, o algoritmo retornava à etapa 2, re-atribuindo os serviços para os papéis. Foram realizadas 600 execuções de cada composição.

O método de seleção dos serviços utilizou os parâmetros de atraso máximo e confiabilidade para excluir os que não atendiam aos requisitos e, dentre os serviços restantes, um era selecionado ao acaso para cumprir o papel. Nos casos onde a redundância era maior que zero, mais de um serviço era selecionado até que se obtivesse a redundância especificada ou que não houvessem mais serviços adequados.

Para as composições criadas, foram coletados os seguintes dados:

1. **Data**: dia, mês e ano em que os serviços da composição começaram a ser executados;
2. **Tempo do envio da requisição**: hora, minuto e segundo do momento em que foi solicitada a realização da composição;
3. **Nome da composição**: qual a composição que foi executada;
4. **Status**: qual foi o resultado final da composição (sucesso ou falha);
5. **Papel que levou à falha**: caso a composição tenha falhado, qual dos papéis foi o responsável;
6. **Lista de serviços utilizados**: lista ordenada de todos os serviços utilizados nesta composição;
7. **Tempo gasto**: tempo, em milissegundos, que a composição levou para finalizar;
8. **Texto de resposta**: resposta recebida pelo último serviço executado na composição.

Os dados de todas as requisições foram tabulados e analisados. As informações extraídas foram: percentual de sucesso e falha da composição como um todo, percentual de falha de cada papel dentro da composição e média e desvio padrão do tempo gasto pelas execuções bem sucedidas das composições.

Na análise dos dados de tempo gasto pelas composições dos grupos, o tempo médio foi comparado usando teste ANOVA de uma via com 95% de confiança. Os resultados estão descritos na subseção seguinte.

6.3 Resultados

Os primeiros resultados obtidos foram os da composição 1A. A contribuição de cada papel para as falhas ocorridas nas requisições desta composição está ilustrada na Figura 6.

O papel que mais ocasionou falhas, proporcionalmente ao número de requisições realizadas, foi o papel 2 (34.47%). Em função disso, construímos as

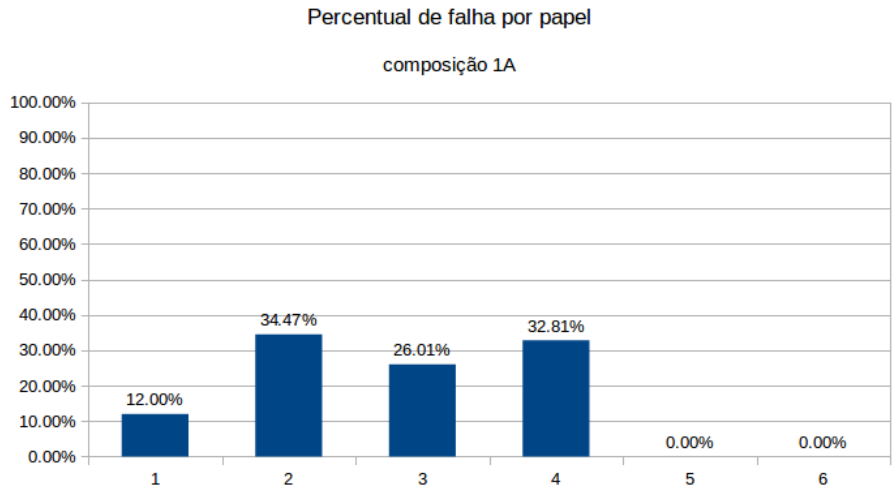

Figura 6: Percentual de falhas causadas por cada papel nas requisições da composição 1A

composições 1B e 1C, com aumento da redundância do papel 2, e as composições 1D e 1E, com aumento da confiabilidade do papel 2.

Os percentuais de sucesso das requisições das composições do grupo 1 estão ilustrados na Figura 7. É possível observar que os melhores resultados foram obtidos com uma maior confiabilidade no papel 2 (composição 1E) e que ambos os tratamentos aplicados (aumento da redundância e da confiabilidade) surtiram efeito na taxa de sucesso. Comparando os resultados de 1A, 1B e 1C temos um aumento de quase 10% na taxa de sucesso e comparando 1A, 1D e 1E esse aumento chega a 12%.

A Figura 8 ilustra a contribuição do papel 2 nas falhas das requisições das composições 1A, 1B e 1C, onde a redundância do papel 2 aumenta de uma composição para a outra (0, 1 e 2). Na Figura 6 temos a mesma informação, agora para as composições 1A, 1D e 1E, em termos da confiabilidade do papel 2 que aumenta (25%, 51%, 76%). Como era de se esperar, há uma relação inversamente proporcional entre as falhas e a redundância e entre as falhas e a confiabilidade.

Os dados de tempo de resposta, para requisições bem sucedidas, levantados para o grupo 1 são mostrados na Figura 10. As barras de erro representam o desvio padrão obtido para cada composição. A variação máxima observada entre os tempos médios foi de 79,00 ms entre as composições 1A e 1C, o que representa uma diminuição de 12,15%. Além disso, o desvio se mostra elevado para todas as composições (menor desvio obtido foi de 261,47 ms na composição 1C).

Realizamos o teste ANOVA de uma via para todas as composições do grupo (1A, 1B, 1C, 1D e 1E), para as composições 1A, 1B e 1C (que representam o

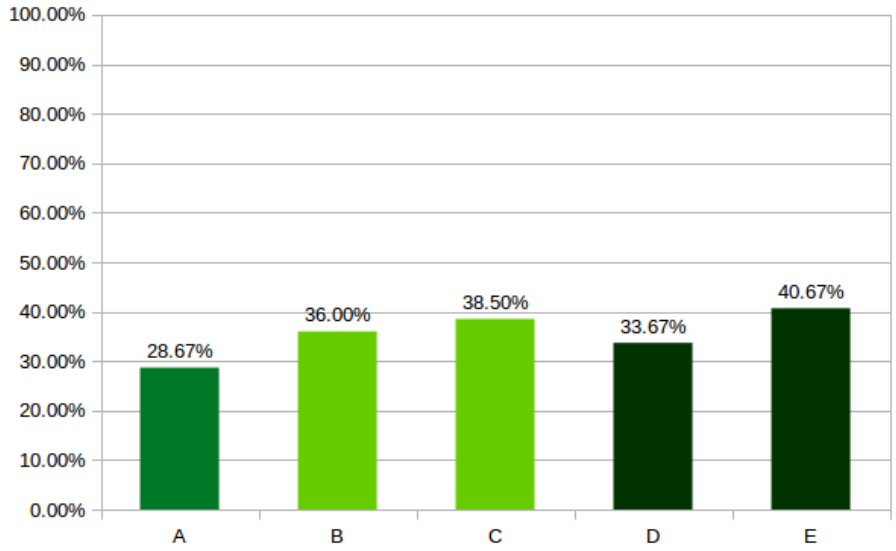

Figura 7: Taxa de sucesso do grupo 1 de composições

primeiro tratamento, com o aumento da redundância do papel 2) e para as composições 1A, 1D e 1E (que representam o segundo tratamento, com o aumento da confiabilidade do papel 2), comparando suas médias de tempo. Os valores obtidos não foram estatisticamente significativos, conforme pode ser observado na Tabela 2. Para que os valores fossem estatisticamente significativos, o F obtido teria que ser maior que o F(a,b) crítico.

Tabela 2: Valores obtidos nos testes ANOVA realizados com as composições do grupo 1

Composições analisadas	F(2,inf) crítico	F(4,inf) crítico	F obtido	Significativo?
1A, 1B, 1C, 1D, 1E	-	2.3719	0.3350504172	Não
1A, 1B, 1C	2.9957	-	0.7149348993	Não
1A, 1D, 1E	2.9957	-	0.6234534691	Não

Para o grupo 2, os dados de sucesso das composições estão na Figura 11. A mudança da redundância dos papéis, de 0 para 1, de forma sequencial (papel 0 na composição 2B, papel 0 e 1 na composição 2C, etc...) ao longo das composições 2A até 2G, teve um efeito nítido na taxa de sucesso. A composição 2A, com redundância zero em todos os papéis, apresentou taxa de sucesso de 43,83%, enquanto a composição 2G, com redundância 1 em todos os papéis apresentou taxa de sucesso mais de duas vezes maior (92%).

Os dados de tempo de resposta das requisições das composições do grupo 2 se encontram na Figura 12. Novamente as barras de erro representam o desvio padrão. Nota-se o aumento do desvio padrão a partir da composição 2D,

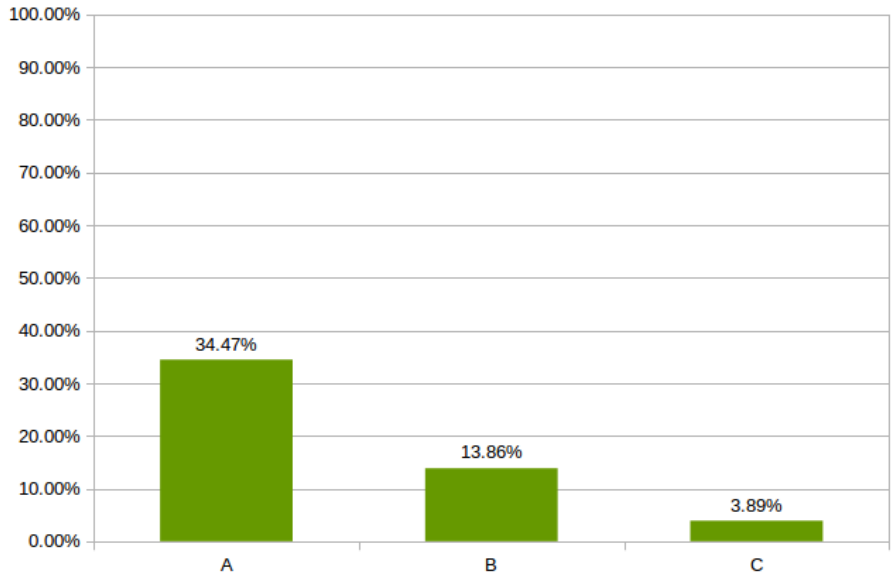

Figura 8: Falhas causadas pelo papel 2 nas composições 1A, 1B e 1C. A diminuição dos valores evidencia o efeito do aumento da redundância do papel 2 na quantidade de falhas causadas pelo mesmo.

atingindo valores maiores do que o valor médio (1111,56 ms para a composição 2G, que possui valor médio de 624,07 ms).

O teste ANOVA de uma via foi feito para a média de todas as composições do grupo 2, de 2A até 2G, e o valor obtido não foi estatisticamente significativo para 5% de significância, já que o valor de F obtido foi menor que F(6,inf). Com base nos dados exibidos na Figura 12, as composições 2A, 2B e 2C foram agrupadas e mais um teste ANOVA foi realizado apenas com estas composições. O motivo do agrupamento foi o decréscimo do valor médio dentro deste grupo. Neste caso o resultado foi estatisticamente significativo para um valor de 5% de significância, já que o valor de F obtido foi maior que F(2,inf). Os valores obtidos estão na Tabela 3.

Tabela 3: Valores obtidos nos testes ANOVA realizados com as composições do grupo 2

Composições analisadas	F(2,inf) crítico	F(6,inf) crítico	F obtido	Significativo?
2A, 2B, 2C, 2D, 2E, 2F, 2G	-	2.0986	0.3350504172	Não
2A, 2B, 2C	2.9957	-	3.1153596282	Sim

As figuras 13 e 14 mostram, respectivamente, os dados de requisições bem sucedidas, em função da diminuição da redundância, e tempo médio de resposta para as composições do grupo 3. Nota-se que só há uma queda expressiva

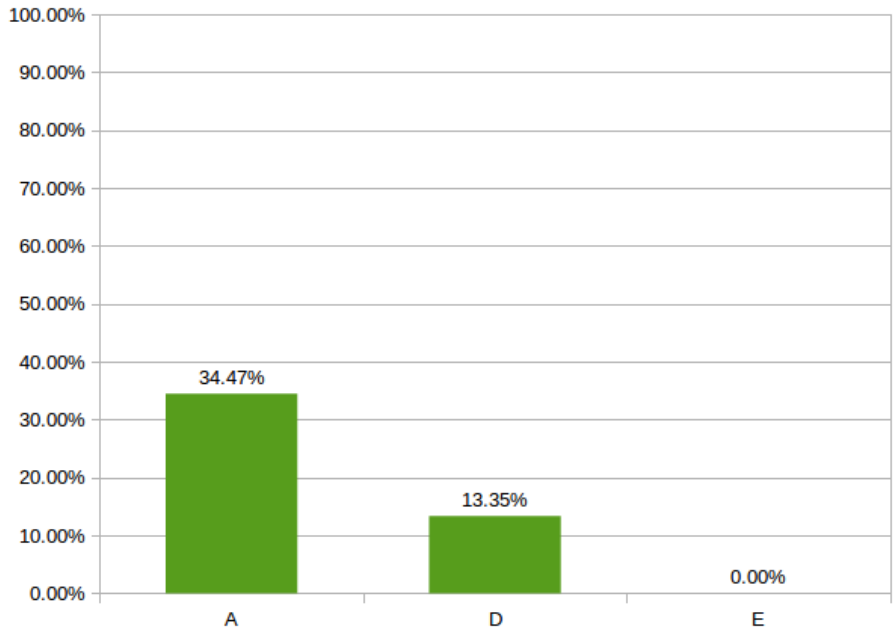

Figura 9: Falhas causadas pelo papel 2 nas composições 1A, 1D e 1E. A diminuição dos valores evidencia o efeito do aumento da confiabilidade do papel 2 na quantidade de falhas causadas pelo mesmo.

(quase 12%) na taxa de sucesso das composições quando a redundância é zero (composição 3F). Nas composições com redundância maior que zero (de 3A até 3E) podemos observar queda na média e desvio padrão do tempo de execução.

O teste ANOVA de uma via foi realizado para todas as composições do grupo 3 porém o valor obtido se mostrou não significativo estatisticamente para 5% de significância, já que o F obtido foi menor que F(5,inf). Com a intenção de avaliar o efeito que uma grande redução na redundância teria sobre o tempo gasto, criamos um subgrupo com as composições 3A, 3D e 3E e realizamos novamente o teste ANOVA de uma via. Desta vez o resultado apresentou significância estatística, já que o F obtido foi maior que F(2,inf). Os dados estão na Tabela 4.

Tabela 4: Valores obtidos nos testes ANOVA realizados com as composições do grupo 3

Composições analisadas	F(2,inf) crítico	F(6,inf) crítico	F obtido	Significativo?
3A, 3B, 3C, 3D, 3E, 3F	-	2.2141	1.489385905	Não
3A, 3D, 3E	2.9957	-	5.3083158326	Sim

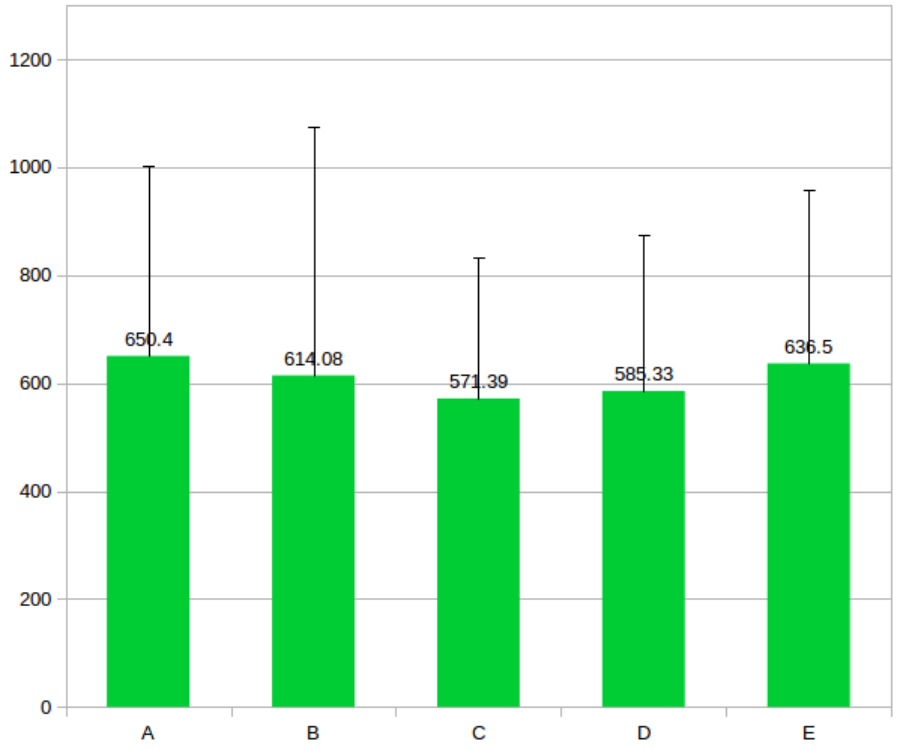

Figura 10: Tempo médio gasto para as composições do grupo 1 bem sucedidas

6.4 Discussão

Para discutir os resultados expostos partiremos da premissa abaixo:

"**Usuários da tecnologia de serviços web que pretendem criar composições de serviços procuram reduzir o uso de recursos (computacionais e financeiros), o tempo de resposta e buscam consistência no desempenho de suas composições.**"

Analisando o tempo médio de resposta no grupo 2 notamos que o aumento da redundância num determinado papel pode reduzir o tempo gasto para se completar uma requisição, como evidenciado pelo resultado do teste ANOVA realizado no subgrupo formado pelas composições 2A, 2B e 2C. No entanto os dados do subgrupo formado pelas composições 3A, 3D e 3E mostram que quando a redundância de um papel aumenta muito, ocorre o efeito contrário.

Possivelmente este aumento do tempo médio é consequência da sobrecarga gerada na máquina realizando as requisições aos serviços, uma vez que foram usadas threads para que várias requisições fossem feitas paralelamente. Por se tratar de uma máquina de uso geral, com diversos outros processos executando concomitantemente às threads criadas para realizar as requisições, os recursos

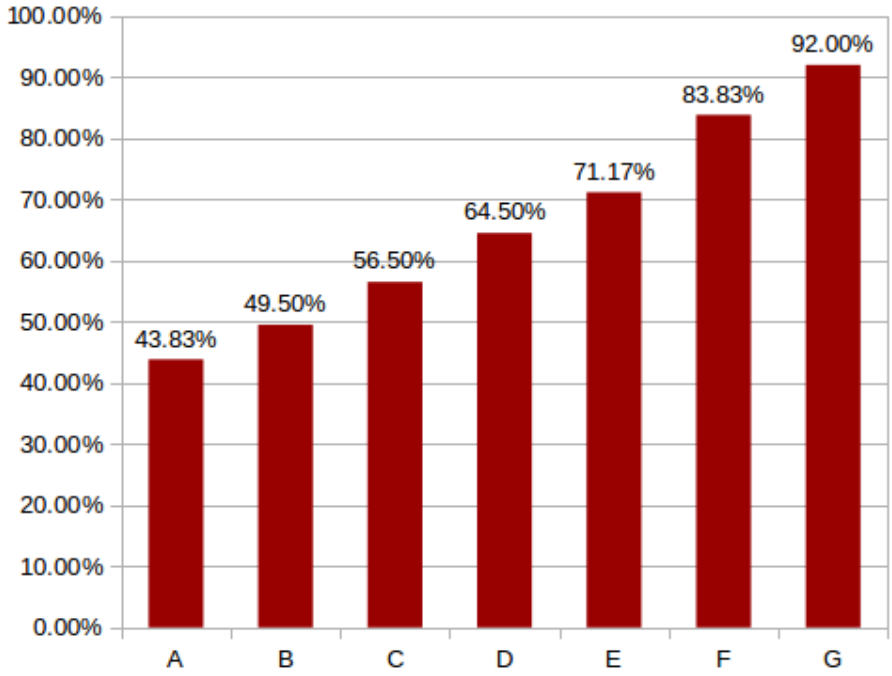

Figura 11: Taxa de sucesso das composições do grupo 2 evidenciando o efeito do aumento da redundância de 0 para 1 em cada papel da composição

computacionais precisam ser compartilhados e isso gera atrasos na execução e prejuízo no desempenho. Uma proposta de trabalho futuro é elaborar experimentos que avaliem o real impacto de várias threads sendo executadas em paralelo durante a invocação de serviços redundantes em uma composição.

O desempenho do grupo 2, em termos de sucesso nas requisições das composições, evidencia os benefícios de não depender apenas de 1 serviço para cumprir certo papel. Com o aumento da redundância dos papéis de 0 para 1, a melhora foi significativa (as requisições que foram bem sucedidas mais que dobraram, passando de 43,83% para 92%). Num cenário real, onde o desenvolvedor da composição não pode modificar os serviços por ela utilizados, aumentando, por exemplo, sua confiabilidade, a estratégia de paralelizar requisições para cumprir papéis críticos da composição é uma boa opção. Os dados mostrados na Figura 5, que mostram o efeito do aumento da redundância do papel 2 nas falhas da composição, corroboram esta ideia. Com apenas uma requisição a mais sendo feita para cumprir o papel 2, a quantidade de falhas causadas por este papel cai mais de 20% (de 34,47% para 13,86%) e com duas requisições a mais a redução nas falhas provocadas pelo papel 2 é de mais de 30% (de 34,47% para 3,89%). Por outro lado, como mostrado na Figura 9, a variação no tempo gasto para se concluir uma requisição com redundância em vários papéis aumenta muito,

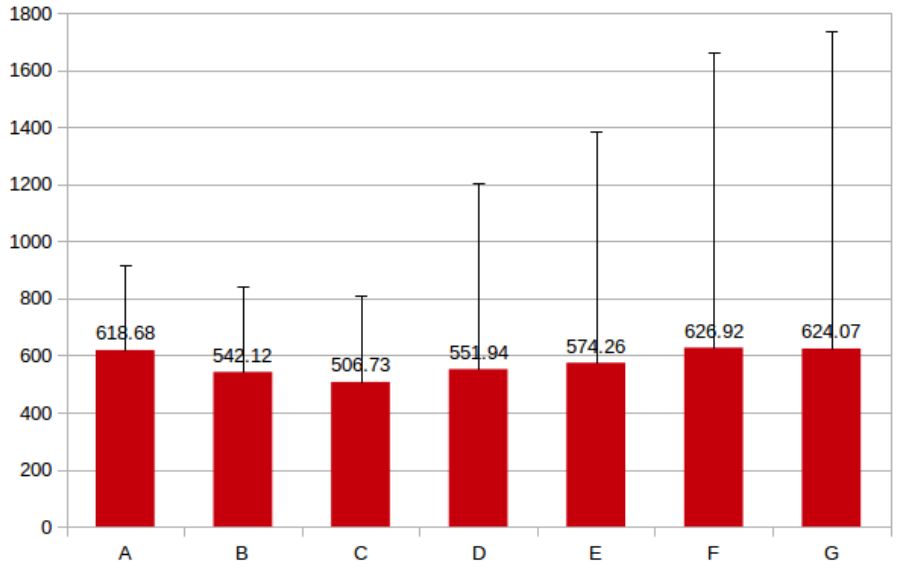

Figura 12: Tempo médio gasto para as composições do grupo 2 bem sucedidas

apresentando um desvio padrão de 1111,56 milissegundos para uma média de apenas 624,07 milissegundos. Este tipo de comportamento errático do tempo de resposta pode ser indesejado, exigindo que uma análise empírica das composições que se pretende criar seja feita para avaliar se este 'efeito colateral' compensa o ganho na taxa de requisições bem sucedidas.

O efeito do aumento no requisito de confiabilidade num papel 'gargalo', como o papel 2 das composições do grupo 1, se mostra eficiente para melhorar a taxa de sucesso das requisições. No entanto, em cenários reais pode não ser possível exigir maior confiabilidade dos serviços necessários, portanto esta estratégia nem sempre é implementável.

Tomando as composições com redundância zero em todos os papéis, 1A, 2A e 3F, temos respectivamente 28,67%, 43,83% e 88,17% de taxas de sucesso. É esperado que a composição 3F apresente maior taxa de sucesso por conter apenas 1 papel, mas a diferença nos resultados das composições 1A e 2A evidenciam a sensibilidade das composições aos requisitos dos papéis que as compõem. A tecnologia dos serviços web, por seu caráter modular, permite grande flexibilidade na hora de construir composições mas, por depender da implementação de serviços criados por terceiros, o comportamento destas requisições torna-se de difícil predição.

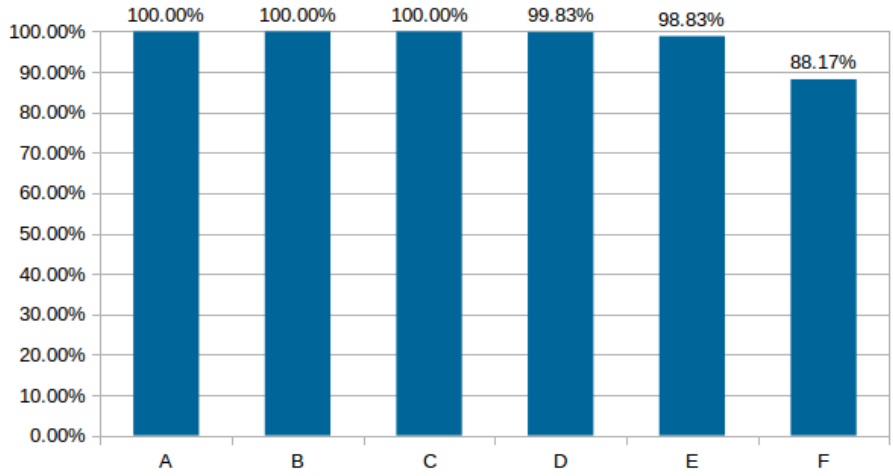

Figura 13: Efeito da diminuição da redundância para o sucesso da composição, visível na taxa de sucesso das composições do grupo 3

7 Conclusões

Tempo médio de resposta num dado período de tempo, percentual de falhas completas num dado período e percentual de falhas parciais são exemplos de requisitos não funcionais relevantes para a escolha de determinado serviço, em particular quando a intenção é montar composições em série, onde um serviço depende da resposta de outro pra terminar sua computação pretendida.

Com os resultados obtidos por meio dos experimentos, percebe-se que não é tarefa simples otimizar o desempenho de uma composição. As composições mostram comportamento variado a depender de sua estrutura de papéis e requisitos não funcionais. No contexto real, algumas estratégias são inviáveis se levarmos em conta nossa premissa de gasto mínimo de recursos e tempo de resposta. Muitas vezes para se obter uma melhora na taxa de sucesso é preciso gastar mais tempo, sendo preciso comprometer algum dos requisitos da composição.

8 Bibliografia

Referências

[1] `https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice` (fevereiro 2016)

[2] `http://www.tutorialspoint.com/webservices/what_are_web_services.htm` (fevereiro 2016)

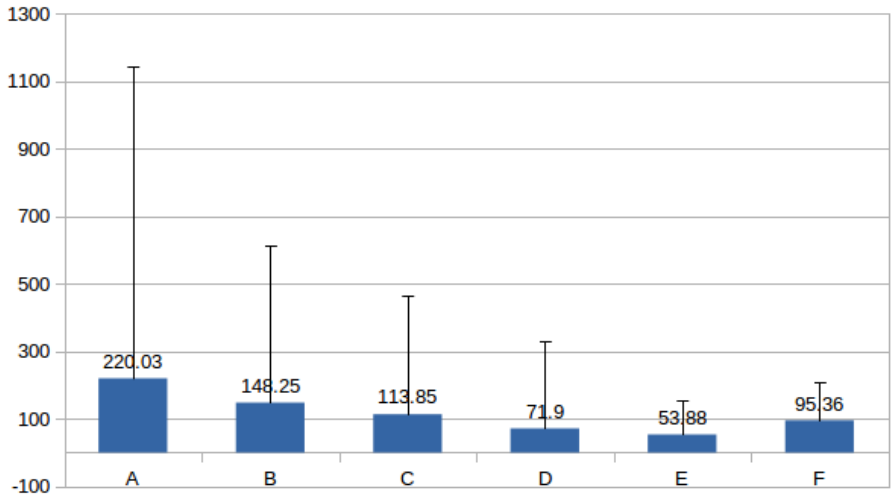

Figura 14: Tempo médio gasto para as composições do grupo 3 bem sucedidas.

[3] Glinz, Martin. "On non-functional requirements." *15th IEEE International Requirements Engineering Conference (RE 2007)*. IEEE, 2007.

[4] Davis, Alan M. *Software requirements: objects, functions, and states.* Prentice-Hall, Inc., 1993.

[5] Kotonya, Gerald, and Ian Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[6] `http://serviceorientation.com/soaglossary/service_composition` (abril 2016)

[7] Sheng, Quan Z., et al. "Web services composition: A decade's overview." *Information Sciences* 280 (2014): 218-238.

[8] Tsesmetzis, Dimitrios T., et al. "QoS awareness support in Web-Service semantics." *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*. IEEE, 2006.

[9] Dobson, Glen, Russell Lock, and Ian Sommerville. "Quality of service requirements specification using an ontology."(2005).

[10] Srivastava, Biplav, and Jana Koehler. "Web service composition-current solutions and open problems." *ICAPS 2003 workshop on Planning for Web Services*. Vol. 35. 2003.

[11] `https://developers.facebook.com/docs/graph-api/overview/` (abril 2016)

[12] https://dev.twitter.com/rest/public (abril 2016)

[13] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (abril 2016)

9 Anexo 1

Tabela 5: Descrição completa das composições do grupo 1 criadas para os experimentos com composições. Destacadas em verde as linhas que definem o primeiro papel de cada composição.

Composição	Número de papéis	Papel	Ação	Redundância	Atraso máximo	Confiabilidade
1A	6	1	random	0	747242905	63%
		2	oposto	0	767976590	25%
		3	div	0	486450975	48%
		4	soma	0	63494004	5%
		5	mod2	0	278288922	81%
		6	mult	0	647780023	76%
1B	6	1	random	0	747242905	63%
		2	oposto	1	767976590	25%
		3	div	0	486450975	48%
		4	soma	0	63494004	5%
		5	mod2	0	278288922	81%
		6	mult	0	647780023	76%
1C	6	1	random	0	747242905	63%
		2	oposto	2	767976590	25%
		3	div	0	486450975	48%
		4	soma	0	63494004	5%
		5	mod2	0	278288922	81%
		6	mult	0	647780023	76%
1D	6	1	random	0	747242905	63%
		2	oposto	0	767976590	51%
		3	div	0	486450975	48%
		4	soma	0	63494004	5%
		5	mod2	0	278288922	81%
		6	mult	0	647780023	76%
1E	6	1	random	0	747242905	63%
		2	oposto	0	767976590	76%
		3	div	0	486450975	48%
		4	soma	0	63494004	5%
		5	mod2	0	278288922	81%
		6	mult	0	647780023	76%

Tabela 6: Descrição completa das composições do grupo 2 criadas para os experimentos com composições. Destacadas em verde as linhas que definem o primeiro papel de cada composição.

Composição	Número de papéis	Papel	Ação	Redundância	Atraso máximo	Confiabilidade
2A	6	1	`random`	0	117756553	59%
		2	`random`	0	117756553	59%
		3	`random`	0	117756553	59%
		4	`random`	0	117756553	59%
		5	`random`	0	117756553	59%
		6	`random`	0	117756553	59%
2B	6	1	`random`	1	117756553	59%
		2	`random`	0	117756553	59%
		3	`random`	0	117756553	59%
		4	`random`	0	117756553	59%
		5	`random`	0	117756553	59%
		6	`random`	0	117756553	59%
2C	6	1	`random`	1	117756553	59%
		2	`random`	1	117756553	59%
		3	`random`	0	117756553	59%
		4	`random`	0	117756553	59%
		5	`random`	0	117756553	59%
		6	`random`	0	117756553	59%
2D	6	1	`random`	1	117756553	59%
		2	`random`	1	117756553	59%
		3	`random`	1	117756553	59%
		4	`random`	0	117756553	59%
		5	`random`	0	117756553	59%
		6	`random`	0	117756553	59%
2E	6	1	`random`	1	117756553	59%
		2	`random`	1	117756553	59%
		3	`random`	1	117756553	59%
		4	`random`	1	117756553	59%
		5	`random`	0	117756553	59%
		6	`random`	0	117756553	59%
2F	6	1	`random`	1	117756553	59%
		2	`random`	1	117756553	59%
		3	`random`	1	117756553	59%
		4	`random`	1	117756553	59%
		5	`random`	1	117756553	59%
		6	`random`	0	117756553	59%
2G	6	1	`random`	1	117756553	59%
		2	`random`	1	117756553	59%
		3	`random`	1	117756553	59%
		4	`random`	1	117756553	59%
		5	`random`	1	117756553	59%
		6	`random`	1	117756553	59%

Tabela 7: Descrição completa das composições do grupo 1 criadas para os experimentos com composições. Destacadas em verde as linhas que definem o primeiro papel de cada composição.

Composição	Número de papéis	Papel	Ação	Redundância	Atraso máximo	Confiabilidade
3A	1	1	`random`	5	117756553	59%
3B	1	1	`random`	4	117756553	59%
3C	1	1	`random`	3	117756553	59%
3D	1	1	`random`	2	117756553	59%
3E	1	1	`random`	1	117756553	59%
3F	1	1	`random`	0	117756553	59%