Regente: Um Arcabouço para Gerenciamento Eficiente de Orquestrações de Serviços Web*

Cássio J. S. Freire[1], Paulo F. Pires[2], Flávia C. Delicato[2], Maria Luiza M. Campos[1], Luci Pirmez[1], Marcel V. M. Oliveira[2]

[1]Programa de Pós Graduação em Informática (IM/NCE) – Universidade Federal do Rio de Janeiro (UFRJ) - Rio de Janeiro - RJ – Brasil

[2]Departamento de Informática e Matemática Aplicada – Universidade Federal do Rio Grande do Norte (UFRN) - Natal - RN - Brasil.

{cassiojsf,mluiza,luci}@nce.ufrj.br,
{paulo.pires,flavia.delicato,marcel}@dimap.ufrn.br

Abstract. *Web services orchestration can be executed using either centralized or disperse (decentralized) models. Several studies suggest that the disperse model performs better in terms of throughput, scalability and response time. However, such assumption does not always hold since the efficiency of each model depends on variants like the orchestration specification and the organization of nodes that provide the Web services and the orchestration engine in the network. This work presents Regente, an infrastructure that optimizes the execution of Web services orchestrations through strategies for choosing the most suitable execution model based on such variants. To validate the proposed strategies, a set of simulations were carried on with different orchestration specifications and network topologies.*

Resumo. *Orquestrações de serviços Web podem ser executadas através de dois modelos diferentes: centralizado e disperso (descentralizado). Vários trabalhos apontam que o modelo disperso oferece um melhor desempenho em termos de throughput, escalabilidade e tempo de reposta. Contudo, esta afirmação não é sempre válida já que a eficiencia de cada modelo depende de variantes como a especificação da orquestração e a organização dos nós que fornecem os serviços Web e a máquina de orquestração. Este trabalho apresenta Regente, um arcabouço que otimiza a execução de orquestrações de serviços Web através de estratégias para seleção do modelo de execução mais adequado baseado nessas variantes. Para validar as estratégias propostas, realizaram-se simulações com diferentes combinações de especificações de orquestração e organizações dos nós.*

1. Introdução

A tecnologia de serviços Web (SW) é atualmente o paradigma mais difundido para prover computação orientada a serviços, fomentando o desenvolvimento de aplicações distribuídas na Internet, as quais requerem um baixo grau de acoplamento e um alto grau de interoperabilidade. Serviços Web, de forma isolada, podem não atender determinados objetivos de negócio. No entanto, é possível que uma composição destes venha prover novas funcionalidades, que agreguem valor ao negócio [Peltz 2003]. Uma composição de serviços

* Trabalho parcialmente financiado pelo CNPq (projetos: 477226/2007-8, 311454/2006-2 e 551210/2005-2) e pela FAPESB (projeto 19.571.216.3383).

Web, também conhecida como orquestração de serviços Web (OSW), pode ser representada por uma especificação (workflow), a qual define um conjunto de atividades, que representam operações dos SW, e a ordem de execução destas, de modo que estas atividades têm entre si um relacionamento de precedência bem definido [Kim and Han 2001].

Especificações de orquestração são instanciadas, executadas e monitoradas por sistemas de gerenciamento de SW. Uma das tarefas do sistema de gerenciamento, relacionada com a execução da orquestração de SW, consiste na alocação de cada atividade da especificação aos SW responsáveis por sua realização, gerando um plano de execução [Yang and Papazoglou 2004]. Um plano de execução corresponde a uma instância de execução de uma especificação de OSWs. A gerência do plano de execução de uma OSW é realizada por uma máquina de orquestração apropriada para a linguagem na qual foi especificada a orquestração [Peltz 2003] e, por definição, sempre é efetuada através de um modelo de execução centralizado. No âmbito de SW, a eficiência do modelo de execução centralizado é comprometida devido a: (i) sobrecarga do nó responsável pela gerência, quando, por exemplo, muitos clientes invocam a orquestração simultaneamente; (ii) indisponibilidade do nó responsável pela gerência em caso de falhas; (iii) sobrecarga dos enlaces próximos ao nó responsável pela gerência, gerando filas de espera; e (iv) características do canal de comunicação, como largura de banda e outras, que influenciam no tempo de envio e recebimento das mensagens [Muth et al. 1998].

Com o objetivo de otimizar a gerência de execução de uma OSW, o modelo disperso (também conhecido como descentralizado) vem sendo proposto como alternativa ao centralizado. Em tal modelo, a gerência do plano de execução é descentralizada. Para prover essa descentralização, a especificação da orquestração é particionada em vários planos de execução e esses são encaminhados e executados em diferentes nós os quais devem possuir uma máquina de orquestração. Vários trabalhos [Nanda and Karnik 2003, Nanda, Chandra and Sarkar 2004, Binder, Constantinescu, and Faltings 2006] apontam que o modelo disperso oferece um melhor desempenho em relação ao centralizado em termos de throughput, escalabilidade e tempo de reposta. Contudo, esta afirmação não é sempre válida já que a eficiência de cada modelo depende de variantes como a especificação da orquestração e a organização dos nós que fornecem os serviços Web e a máquina de orquestração na rede.

Este trabalho propõe o **Regente,** um arcabouço para a gerência eficiente de orquestrações de SW. Gerenciar de forma eficiente OWSs é uma necessidade essencial da computação orientada a serviços, já que o resultado global da execução de uma orquestração afeta diretamente a percepção do consumidor do serviço. Ou seja, a eficiência na execução de uma orquestração influencia a QoWS (*Quality of Web Service*) [Ouzzani and Bouguettaya 2004], impactando no resultado esperado pelo consumidor.

O processo de gerência do Regente visa otimizar a execução de orquestrações de SW através de estratégias para a seleção do modelo de execução mais adequado de acordo com as características da especificação da orquestração e da organização dos nós que fornecem os serviços Web e a máquina de orquestração na rede. Para realizar a seleção do modelo de execução foi proposto um algoritmo concebido a partir de relatos da literatura e ajustado através de simulações. O presente artigo descreve o arcabouço proposto, englobando, os serviços disponibilizados, as estratégias propostas para a otimização da gerência e as simulações realizadas.

O restante deste artigo está organizado como segue: a Seção 2 categoriza os modelos de gerência de orquestração de serviços; as Seções 3 e 4 descrevem o Regente e detalham as

estratégias usadas por ele para a gerência eficiente. Na Seção 5 são descritas as simulações e seus resultados. A Seção 6 apresenta os trabalho relacionados e a Seção 7 conclui o trabalho.

2. Modelos de Execução de Orquestração de SW

Um sistema Web pode ser representado por um grafo que consiste de um conjunto de vértices, representando os nós $N = \{N_1, N_2,..., N_n\}$, conectados por arestas, representando os canais de comunicação entre eles. No contexto deste trabalho, os nós podem ser classificados em nós de serviço, execução, híbrido ou origem. Entende-se por nós de serviços $Ns \in N$ aqueles nós que disponibilizam apenas os SW. Nós de controle $Nc \in N$ disponibilizam instâncias da máquina de orquestração e são responsáveis por efetuar o gerenciamento do plano de execução da orquestração. Já os nós híbridos $Nh \in N$ são nós que atuam tanto no papel de nó de serviço, ao disponibilizar SW, como no papel de nó de controle, ao fornecer uma instância da máquina de orquestração. Nós de origem $No \in N$ são nós responsáveis pela descoberta dos nós de serviço e nós de controle para uma dada topologia de rede.

Cada par de nós (N_i, N_j) está associado a um canal Cc com capacidade de comunicação Cm_{ij}, a qual indica o volume de dados que pode ser transmitido do nó N_i para o nó N_j em uma unidade de tempo. A capacidade de comunicação originada de um nó para ele mesmo é representada por Cm_{ii}. Quanto ao grau de descentralização da gerência de orquestração, os modelos de execução discutidos no presente trabalho são classificados em centralizado e disperso [Kim and Han 2001]. As próximas seções detalham estes modelos.

2.1 Modelo Centralizado

A orquestração centralizada é o modelo mais tradicional para a execução de OSW. Nele, a gerência de execução da orquestração é de responsabilidade de um único nó de controle Nc. Como um único nó possui o controle completo sobre a orquestração, tarefas de monitoramento e de gerenciamento são facilitadas, sendo essa uma das vantagens desse modelo [Muth et al. 1998]. Por outro lado, sua principal desvantagem é que o encaminhamento de mensagens pode ser feito de forma ineficiente, já que nem sempre as mensagens de dados são enviadas diretamente do SW que produz o dado para o que consome o dado. Além disso, devido ao fato da gerência ficar sob a responsabilidade de um único nó, outras deficiências inerentes a arquiteturas centralizadas podem acontecer, como a ocorrência de falhas decorrentes de indisponibilidades do nó de controle e a sobrecarga de processamento no mesmo.

2.2 Modelo Disperso

Em um modelo disperso, segundo Kim and Han (2001), o controle da execução e os dados da aplicação são divididos em múltiplas partições, que são mantidas em diferentes nós de forma não replicada. Portanto, na orquestração dispersa, a especificação da orquestração é **particionada** em vários planos de execução e esses são encaminhados e executados em diferentes nós de controle Nc. Na orquestração dispersa, é possível reduzir o tamanho e o número de mensagens trafegadas na rede e ainda escolher as rotas de comunicação que não comprometam a QoWS efetivamente oferecida [Chafle et al. 2004]. Nesse modelo, as mensagens podem ser encaminhadas diretamente do SW que produz o dado para o SW que consome o dado. As vantagens da orquestração dispersa estão relacionadas com as mesmas questões inerentemente resolvidas através da distribuição, como o balanceamento de carga e a tolerância a falhas [Nanda and Karnik 2003], além de possibilitar que informações sobre as características dinâmicas do ambiente em tempo de execução e da topologia da rede possam ser usadas para decidir o melhor particionamento a ser aplicado e, por conseguinte, para

obter uma orquestração eficiente. A principal desvantagem do modelo disperso é a complexidade adicional requerida para assegurar a adequada coordenação entre os nós.

Um caso particular do modelo disperso, conhecido como modelo **descentralizado** [Kim and Han 2001], ocorre quando todos os nós de serviço Ns_i são nós híbridos Nh_i, ou seja, todos Ns_i disponibilizam uma instância da máquina de orquestração, na qual as informações da instância de execução da orquestração são replicadas. Uma característica importante desse caso particular do modelo disperso é que o número de mensagens encaminhadas é menor que o número de mensagens de uma execução centralizada e dos outros casos do modelo disperso. Contudo, a exigência de que todo nó de serviço seja híbrido não representa um cenário realista de um sistema Web.

3. Um Arcabouço para Gerenciamento Eficiente de Orquestrações de SW

Neste trabalho é proposto o Regente, um arcabouço para o gerenciamento de orquestrações de SW, com o objetivo de gerar planos de execução eficientes. Para tal, o Regente disponibiliza os serviços de descoberta e de gerenciamento, que apóiam a geração e a otimização de planos de execução, bem como um serviço de execução. O serviço de gerenciamento é o foco do presente artigo e será descrito em maiores detalhes.

3.1 Serviço de Descoberta

O serviço de descoberta do Regente é composto pelos módulos de descoberta de SW e de descoberta de Nc e por um repositório de metadados que contém as informações sobre nós de serviços (incluindo o QoSW provido por eles) e nós de controle disponíveis. O ***módulo de descoberta de SW*** tem como objetivo descobrir os serviços Web apropriados para realizar cada atividade da especificação de uma orquestração. A descoberta é realizada através de uma requisição de consulta submetida ao repositório de metadados. Esta consulta leva em consideração os requisitos funcionais e os critérios de QoWS de cada atividade. Como resultado da consulta é obtida uma lista de serviços disponíveis para cada atividade especificada na orquestração. Essa lista de serviços inclui metadados do serviço como custo monetário, nível de segurança, etc., bem como a sua localização topológica.

O ***módulo de descoberta dos Nc*** envia requisições de consulta ao repositório de metadados com o objetivo de encontrar os *Nc* apropriados para executar a orquestração. Os critérios de busca por nós de controle podem ser: a sua localização, a sua capacidade computacional, a capacidade dos canais de comunicação que os interligam a outros nós.

3.2 Serviço de Gerenciamento

O serviço de gerenciamento implementa um processo composto de 4 etapas: (i) alocar as atividades para os nós de serviços; (ii) escolher modelo de execução; (iii) gerar partições (aplica-se apenas ao modelo de execução disperso); (iv) alocar a especificação da orquestração a nós de controle de acordo com o modelo de execução (escalonamento da orquestração); e (v) gerar plano(s) de execução. Como tais etapas implementam a otimização da gerência da orquestração, foco do presente artigo, elas serão detalhadas na Seção 4.

3.3 Serviço de Execução

O serviço de execução do Regente tem como função instanciar e executar a orquestração de serviços Web de acordo com os planos de execução gerados pelo serviço de gerenciamento. O serviço de execução implementa uma máquina de orquestração que tem a capacidade de interpretar a especificação de uma orquestração, controlar as instâncias de orquestração,

invocar os SW que foram alocados para cada atividade e tratar dados relevantes para a orquestração, como os advindos das mensagens de dados e controle.

4. Gerenciamento Eficiente de Orquestração de SW no Regente

Uma orquestração θ pode ser especificada por um grafo acíclico direcionado (DAG, do inglês *Directed Acyclic Graph*) que é definido como um par (Γ, Λ): o primeiro elemento, Γ, é o conjunto de vértices *A* cujos elementos A_i ($1 \leq i \leq card(\Gamma)$) representam as atividades. O segundo elemento, Λ, é o conjunto de arestas $A_i \rightarrow A_j$ que indicam a dependência entre uma atividade pai A_i e uma atividade filha A_j, a qual não pode ser executada até que todas as suas atividades pai tenham sido completadas. Em um grafo de orquestração, a atividade que não tem atividade pai é denotada $A_{entrada}$ e a atividade que não tem atividade filha é denotada por A_{saida}. Informalmente, podemos definir o conjunto de todas as atividades de entrada ε como o conjunto de todas as atividades A_j de *A* tais que não exista uma atividade A_i ($1 \leq i \leq card(\Gamma)$) que seja pai de A_j, ou seja, $A_i \rightarrow A_j \in \Lambda$. Formalmente, temos que $\varepsilon = \{A_j : A \mid (\neg \exists\, i : 1..card(\Gamma) \bullet A_i \rightarrow A_j \in \Lambda)\}$. A notação matemática de compreensão de conjuntos usada é aquela de Z [Woodcock et al. 1996]: $\{x : T \mid P(x) \bullet F(x)\}$ é o conjunto de todos os $F(x)$ onde x é um elemento de T que satisfaz a condição $P(X)$. De maneira análoga a ε, temos que o conjunto de todas as atividades de saída σ é formado por todas as atividades A_i que não possuem filhos em Λ, ou seja, $\sigma = \{A_i : A \mid (\neg \exists\, j : 1..card(\Gamma) \bullet A_i \rightarrow A_j \in \Lambda)\}$. Uma orquestração é construída através da ligação de múltiplas atividades *A* de acordo com as suas dependências.

Dada uma orquestração θ, uma topologia *T* e o custo de execução da orquestração, $Cosw_\theta$, o problema de gerenciamento eficiente de orquestração tratado pelo Regente visa:

1. Alocar exatamente um nó de serviço *Ns* para cada uma das atividades em *A*, obedecendo os requisitos funcionais e não funcionais (QoWS) estabelecidos pelo usuário;
2. Definir um modelo de execução *ME* para a orquestração θ de acordo com a topologia *T* corrente (*ME*(θ,*T*)) tal que $Cosw\theta_{ME}$ seja minimizado; ou seja, sendo η o conjunto de todos os modelos de execução, não existe outro modelo *ME*' de execução de η tal que o custo $Cosw\theta_{ME'}$ seja menor que $Cosw\theta_{ME}$ ($\neg \exists\, ME' : \eta \bullet Cosw\theta_{ME'} < Cosw\theta_{ME}$);
3. Gerar um conjunto *PE* de planos de execução de acordo com o *ME* escolhido;
4. Para todo plano de execução *P* em *PE*, alocar um nó de controle Nc_j da topologia *T* tal que: (ME=centralizado⇒j=1) ∧ *ME=disperso*⇒*j*:1..*card*(Γ) ou seja, caso o modelo de execução seja centralizado, temos a alocação do nó de controle Nc_1; caso contrário deve-se alocar o nó de controle Nc_j tal que *j* seja um número entre 1 e o número de atividades em Γ.

$Cosw_\theta$ é função de uma série de características dos nós e dos canais de comunicação *Cc*, já citadas neste trabalho. Cada nó *N*, seja de serviço ou de controle, possui uma capacidade computacional *Ccap*(*N*). Da mesma forma, cada canal de comunicação *Cc* entre um nó N_i e um nó N_j tem a sua capacidade de comunicação Cm_{ij}. Para efeito de

simplificação, considera-se nesse trabalho que todos os nós têm a mesma capacidade computacional *Ccap* ($\forall i, j \mid i \neq j \bullet Ccap(N_i) = Ccap(N_j)$).

No escopo deste trabalho, *Cosw* é representado pelo somatório do custo de processamento *Tprocess* e de comunicação *Ccom* de todos os nós de serviço *Ns* especificados em um plano de execução *P*. Para fins de simplificação, considera-se que *Tprocess* engloba também o tempo em fila ocasionado pela carga do nó onde o SW é implantado. Além disso, segundo Ozsu and Valduriez (1999), o custo de comunicação *Ccom* pode ser dado pela fórmula $Ccom=Tmsg_i+(Tr_i*Nb_i)$ onde: *Tmsg* é o custo fixo de inicialização de uma mensagem, *Ttr* o tempo que se leva para transmitir uma unidade de dados de um nó para outro e *Nb* a quantidade de bytes a serem transmitidos na mensagem. Logo, conclui-se que o custo de execução da orquestração pode ser obtido através da fórmula: $Cosw = \sum_{i=1}^{n} T\,\mathrm{Pr}ocess(Ns_i) + \sum_{k=1}^{l} Ccom_k$, onde *n* é número de nós de serviço e *l* é o número de mensagens da orquestração.

As próximas seções detalham as etapas incluídas no gerenciamento do Regente.

4.1 Escalonar Atividades

O escalonamento de atividades é gerado por um algoritmo que recebe como entradas: (i) a lista *L* de serviços Web Ns_k ; (ii) a Topologia *T* de comunicação; ambas retornadas pelo serviço de descoberta; e (iii) a especificação de orquestração θ. O objetivo desse algoritmo consiste em selecionar, dentre a lista *L* de serviços Web candidatos, aquele serviço que melhor se adequa a cada atividade da orquestração, de forma que os critérios de QoWS que abrangem toda a orquestração sejam satisfeitos ao mesmo tempo em que o custo *Cosw* é minimizado (considerando a topologia *T*). A saída do algoritmo é um um vetor esc_a de pares (A_i, Ns_k), onde o primeiro elemento é uma atividade da orquestração θ e o segundo elemento é um nó de serviço em *L*. Na versão atual do Regente, o algoritmo se reduz a associar cada atividade a exatamente um nó de serviço, ou seja, encontrar A_i e Ns_k, tal que *i*=*k*.

4.2 Escolher Modelo de Execução

A escolha do modelo de execução de orquestração é feita por um algoritmo que recebe como entradas: (i) o escalonamento de atividades esc_a; (ii) a lista *M* de nós de controle candidatos; (iii) a topologia de rede *T;* (iv) a especificação de orquestração θ. O objetivo desse algoritmo é escolher o modelo de execução *ME* mais adequado para minimizar o *Cosw*. Além do modelo, o algoritmo também inclui as decisões sobre a estratégia de particionamento, ou seja, sobre o tipo de modelo disperso a ser utilizado. Os tipos de modelo disperso empregados no Regente (Mn1 e Mn2) são descritos na (Seção 4.3.1). A escolha do modelo de execução é considerada um problema NP-Difícil [Canfora et al. 2005]. Assim, o Regente emprega um algoritmo aproximado para solucionar esse problema. Esse algoritmo (Figura 1) foi especificado com base nas características de cada modelo de execução, sendo que seus parâmetros (aqueles que definem os limites das funções indicadas nas linhas 12, 14, 15 e 18 do algoritmo) foram ajustados através de extensivas simulações, descritas na Seção 5. Os parâmetros considerados como influenciadores no desempenho dos modelos de execução foram: (i) grau de dispersão dos nós de serviços; (ii) distância entre os nós de serviço e nós de controle; (iii) número de atividades da orquestração; (iv) tipos de dependência entre as atividades; (v) presença de nós híbridos.

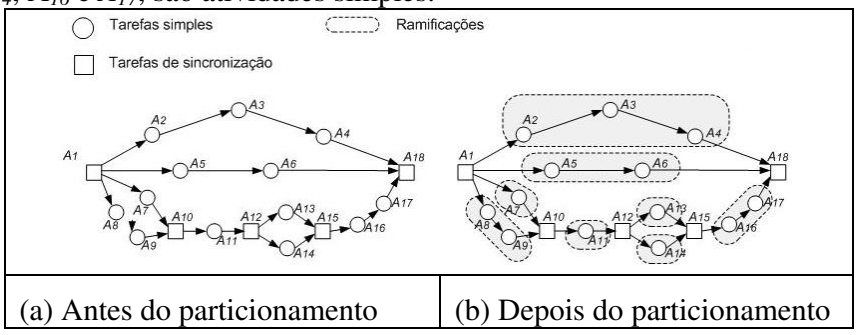
Em um abuso de notação nós utilizamos a notação de compreensão de listas muito próxima a de Haskell [Hutton, 2007] para definir os vetores calculados dentro do algoritmo. De uma maneira geral, $\langle x : T \mid P(x) \bullet F(x) \rangle$ é a lista de todos os $F(x)$ onde x é um elemento de T que satisfaz a condição $P(X)$.

1	**Entrada:** esc_a, M, T, θ
2	**Saída:** ME
3	$(\Gamma, \Lambda) \leftarrow \theta$
4	$numAtividades \leftarrow card(\Gamma)$
5	$vetorNs \leftarrow \langle Ns_k, Ns_i : T \mid (\exists A_k, A_i : A \bullet \{(Ns_k, A_k), (Ns_i, A_i)\} \subseteq esc_a \ \wedge \ A_k \rightarrow A_i \in \Lambda) \bullet (Ns_k, Ns_i) \rangle$
6	$saltosNs \leftarrow \langle saltos(Ns_k, Ns_i) \mid (Ns_k, Ns_i) \ \in \ vetorNs \rangle$
7	$dispersaoNs \leftarrow media(saltosNs)$
8	$vetorDistNc \leftarrow \langle Ns_i : esc_a, Nc_j : M \mid (\forall Nc_k : M \bullet saltos(Ns_i, Nc_j) \leq saltos(Ns_i, Nc_k)) \bullet saltos(Ns_i, Nc_j)) \rangle$
9	$dispersaoNc \leftarrow media(vetorDistNc)$
10	SE $\forall Ns_i : esc_a \bullet hibrido(Ns_i)$ ENTÃO
11	$ME \leftarrow$ *modelo disperso Mn1*
12	SENÃO SE $pequeno(numAtividades)$ ENTÃO
13	$ME \leftarrow$ *modelo centralizado*
14	SENÃO SE $medio(numAtividades)$ ENTÃO
15	SE $baixa(dispersaoNc)$ ENTÃO
16	$ME \leftarrow$ *modelo disperso Mn1*
17	SENÃO $ME \leftarrow$ *modelo centralizado*
18	SENÃO SE $alta(dispersaoNs)$ ENTÃO
19	$ME \leftarrow$ *modelo disperso Mn2*
20	SENÃO SE
21	$ME \leftarrow$ *modelo disperso Mn1*
22	RETORNA ME

Figura 1. Algoritmo para escolha do modelo de orquestração

4.3 Agrupar Atividades em Partições

A próxima etapa, no caso de escolha do modelo de execução disperso, é a definição da forma de particionamento das atividades dentro de uma orquestração. O particionamento de atividades é feito com base nas dependências que existem entre as atrividades de uma orquestração. Assim, as atividades A_i de uma orquestração θ são categorizadas em: (i) **atividades simples**, que possuem uma e somente uma atividade pai e uma e somente uma filha; (ii) **atividades de sincronização**, que possuem mais de uma atividade pai e/ou atividade filha. Na Figura 2a, A_1, A_{10}, A_{12}, A_{15} e A_{18} são atividades de sincronização e $A_2 - A_9$, A_{11}, A_{13}, A_{14}, A_{16} e A_{17}, são atividades simples.



(a) Antes do particionamento | (b) Depois do particionamento

Figura 2. Particionamento da orquestração

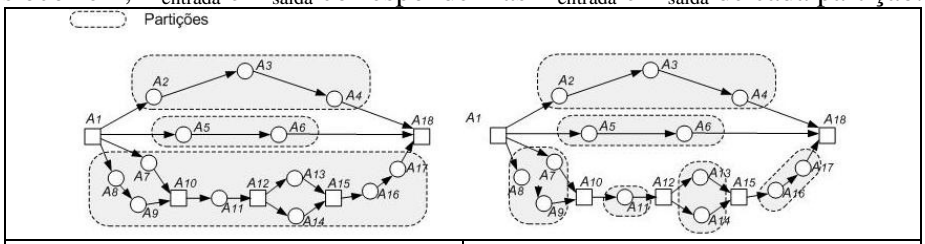
Define-se uma ramificação como um conjunto de atividades simples executadas seqüencialmente entre duas atividades de sincronização. Por exemplo, as ramificações da Figura 2b são: $\{A_2, A_3, A_4\}$, $\{A_5, A_6\}$, $\{A_7\}$, $\{A_8, A_9\}$, $\{A_{11}\}$, $\{A_{13}\}$, $\{A_{14}\}$ e $\{A_{16}, A_{17}\}$. Uma orquestração θ_i pode ser particionada em ramificações independentes R_i $(1 \leq i \leq k)$ e atividades de sincronização Y_i $(1 \leq i \leq l)$, de forma que k e l correspondem, respectivamente, ao número total de ramificações e ao número total de atividades de sincronização da orquestração. Assim, uma orquestração θi composta por n atividades pode ser particionada como $\theta i = (\theta p_1, \theta p_2, ..., \theta p_m)$, onde $1 < m \leq n$ e θp são subgrafos válidos de θ.

4.3.1 Tipos de Particionamento

O Regente emprega duas estratégias de particionamento: (i) completo e (ii) multinível. A decisão entre utilizar uma ou outra estratégia é realizada pelo algoritmo (Figura 1).

Um particionamento é considerado completo quando as partições não incluem nenhuma atividade de sincronização. Ou seja, nesse particionamento uma partição nada mais é que uma ramificação de atividades seqüenciais entre duas atividades de sincronização. Por exemplo, a especificação da orquestração da Figura 2a, gerou 8 partições ilustradas na Figura 2b. O particionamento completo é considerado máximo, já que não existem variações da forma completa, pois a orquestração já não pode ser adicionalmente particionada.

Quanto ao particionamento multinível (Mn), o agrupamento de atividades pode ocorrer recursivamente. Novos particionamentos multinível ocorrerão sucessivamente até que o agrupamento de atividades não inclua nenhuma atividade de sincronização (exceto $A_{entrada}$ e A_{saida}) obtendo, assim, ramificações de atividades seqüenciais entre $A_{entrada}$ e A_{saida}, o que caracteriza o último nível do particionamento. Caso os agrupamentos de atividades em cada partição contenham atividades de sincronização diferentes de $A_{entrada}$ e A_{saida} da orquestração original, caracteriza-se um agrupamento multinível de nível intermediário. Quando o particionamento multinível é aplicado pela primeira vez (nível 1), as atividades $A_{entrada}$ e A_{saida} são as mesmas da orquestração original e, à medida que novos passos recursivos ocorrem, $A_{entrada}$ e A_{saida} correspondem as $A_{entrada}$ e A_{saida} de cada partição.



(a) Particionamento nível 1 (b) Particionamento nível 2

Figura 3. Particionamento multi-nível

Um exemplo do particionamento multinível é ilustrado na Figura 3. A Figura 3a mostra o nível 1 desse particionamento (Mn1), onde $A_{entrada}$ e A_{saida} correspondem respectivamente a A_1 e A_{18}. Neste nível, são gerados três agrupamentos, que correspondem as partições I $\{A_2$, A_3 e $A_4\}$, II $\{A_5$ e $A_6\}$ e III $\{A_7$, ..., $A_{17}\}$. No particionamento multinível de nível 2 (Mn2), mostrado na Figura 3b, as partições I e II não sofrem novos agrupamentos, pois não possuem atividades de sincronização (além das de entrada e saída). No entanto, a partição III gera quatro novos agrupamentos a partir das atividades de sincronização A_{10}, A_{12} e A_{15}, que correspondem respectivamente a partição III.a $\{ A_7$, A_8 e $A_9\}$, III.b $\{ A_{11}\}$, III.c $\{A_{13}$ e $A_{14}\}$ e III.d $\{A_{16}$ e $A_{17}\}$. Se for aplicado mais um nível de particionamento em III.a, III.c e III.d, novas partições serão geradas e o particionamento obtido será equivalente ao completo, mostrado na Figura 2b.

4.4 Escalonar Orquestração

Tendo-se definido o modelo de execução e, no caso de modelo disperso, os particionamentos da orquestração, tem início a etapa de escalonamento da orquestração. O objetivo desta etapa é gerar e distribuir o plano de execução P otimizado.

O plano de execução P é gerado de acordo com o modelo de orquestração escolhido. Para o modelo de orquestração disperso são gerados *n* Ps, equivalentes aos *n* particionamentos estabelecidos. O próximo passo é escalonar os nós de controle de acordo com os Ps gerados. Em seguida, os Ps serão distribuídos de acordo com esse escalonamento.

As decisões sobre o escalonamento dos nós de controle devem ser fundamentadas nas características do ambiente de execução, de forma que os Ps possam ser distribuídos para nós de controle com características mais propícias à sua execução. No Regente, o parâmetro utilizado na decisão de escalonamento é o custo $Cosw_{\theta}$, derivado, dentre outros fatores, do número de saltos entre os nós de controle e cada partição. Para o modelo centralizado, calcula-se um único somatório de saltos, usando todos os nós de serviços contidos na orquestração e, para o modelo disperso, são calculados somatórios para cada partição.

5. Simulações e Análises

Nessa Seção são descritas as simulações realizadas para avaliar os modelos de orquestração, os tipos de particionamento e as propriedades que afetam seu desempenho. Utilizou-se o simulador de eventos discretos JiST e a biblioteca SWANS, ambos desenvolvidos em Java [Barr, Haas, and Renesse]. As simulações foram executadas em uma máquina baseada em AMD, com sistema operacional Windows XP e com as seguintes configurações de hardware: Athlon XP 2.08 GHz e 1.00 GB de RAM. A métrica de avaliação utilizada nas simulações foi o tempo de resposta, obtido conforme a unidade de tempo simulado JiST. Cada resultado foi obtido pela média de 20 rodadas de simulações.

A topologia de rede utilizada nas simulações foi criada pelo gerador de topologias BRITE [Medina, Matta, and Byers 2001]. Essa topologia constituiu-se de um grafo cujo arranjo é estabelecido de forma aleatória, onde os vértices correspondem aos nós de rede e as arestas aos canais de comunicação que ligam esses nós. Foi gerada uma topologia lógica de rede para 900 nós, segundo o modelo de probabilidade de Waxmam. De acordo com esse modelo, inicialmente os nós são uniformemente distribuídos em um plano e, em seguida, são interconectados segundo probabilidades que dependem da distância entre eles. Após a interconexão dos nós, contrói-se uma matriz adjacente com o número de saltos entre cada nó da rede definido através do menor caminho Dijkstra [Cormen, Leiserson and Rivest 2001].

Para fins de simplificação, os nós de origem não são considerados na simulação; todos os nós de controle têm a mesma capacidade de processamento e todas as requisições feitas à orquestração são atendidas considerando que não há filas nos nós de controle. Cada nó de serviço disponibiliza apenas um tipo de serviço Web. Todos os 900 nós simulados são potencialmente nós de serviço e/ou de controle. O papel de cada nó é definido de acordo com o objetivo de cada simulação. A capacidade dos canais de comunicação varia de 10 a 1.024 bytes. Para o particionamento multinível, foram considerados somente os particionamentos completo, multinível nível 1 (Mn1) e multinível nível 2 (Mn2). O tamanho da mensagem de dados foi fixado em 1024 Bytes e da mensagem que transporta a especificação da orquestração em 256 Bytes. As orquestrações usadas nas simulações foram especificadas como variações seguindo o padrão multimerge [van der Aalst et al. 2003]. Esse padrão pode ser ilustrado como uma parte do workflow da Figura 2a identificado pelas atividades A12 – A15. Esse padrão pode corresponder, por exemplo, a uma aplicação de reservas de pacote de

viagem, onde são representados os serviços de reserva de linha aérea (A12), reserva de hotel (A13), reserva de automóveis (A14) e geração da fatura (A15). Nesse exemplo, os serviços de reserva de hotel e de automóveis precisam aguardar que a reserva de passagens seja feita, da mesma forma que a geração da fatura só pode ocorrer depois que a reserva de hotel e de automóveis forem realizadas, as quais, por sua vez, podem ocorrer em paralelo.

Em todas as simulações realizadas no trabalho, o primeiro passo consiste na geração de uma especificação de orquestração. O segundo passo é fazer o escalonamento de serviços que consiste em alocar cada nó de serviço Ns_i a cada atividade A_i da orquestração. Tal escalonamento resulta em um vetor esc_a de pares (A_i, Ns_i). O próximo passo é a geração das partições (no caso da centralizada considera-se que existe somente uma partição). Para cada partição, atribui-se um nó de controle *Nc*. Cada nó de controle é escolhido com base no menor somatório do número de saltos, considerando todos os nós de serviços alocados para cada partição. O resultado desse passo é um vetor de pares (Nc_j, esc_{ap}), onde esc_{ap} corresponde ao subvetor do vetor de pares esc_a (associado a uma partição) resultante do escalonamento de serviços. No quarto passo é feito o encaminhamento da orquestração para o(s) nó(s) de controle e, finalmente, no quinto passo, é executada a orquestração. No caso do particionamento Mn2, os dois últimos passos são feitos de forma recursiva.

O primeiro conjunto de simulações foi conduzido para avaliar a influência do número de atividades e inter-dependências das mesmas no desempenho dos diferentes modelos de orquestração. No segundo conjunto de simulações buscou-se avaliar a influência da dispersão dos nós de serviços e dos nós de controle no desempenho dos modelos.

5.1 Avaliando o Impacto da Quantidade e Interdependência das Atividades

No primeiro conjunto de simulações, estabeleceu-se uma distância fixa de 4 saltos entre os nós de serviços antecessores e sucessores dentro de uma especificação de orquestração e de 2 saltos entre um nó de serviço qualquer e o nó de controle mais próximo a ele. Cabe ressaltar que, como nessas simulações o objetivo era avaliar a influência das diferentes dependências entre atividades, o número de saltos entre nós foi arbitrariamente fixado nesses valores, já que essa variante não afeta os resultados. Na primeira etapa, foram geradas diferentes especificações de orquestrações seguindo o padrão multimerge com duas atividades de sincronização e apenas duas ramificações, tendo cada uma um número variável de atividades seqüênciais. A interdependência das atividades é definida pela especificação de relacionamentos sequencias ou paralelos entre elas. A Figura 4 ilustra os resultados obtidos. Em uma segunda etapa, foi variado o número de ramificações mantendo o número fixo de 1 (uma) atividade seqüencial em cada ramificação. Os resultados são apresentados na Figura 5.

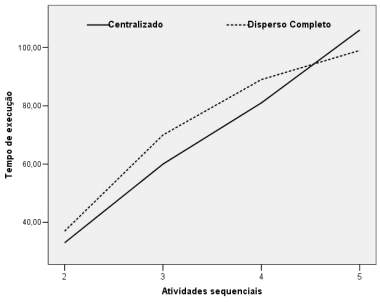


Figura 4. Impacto da variação da quantidade e interdependência das atividades

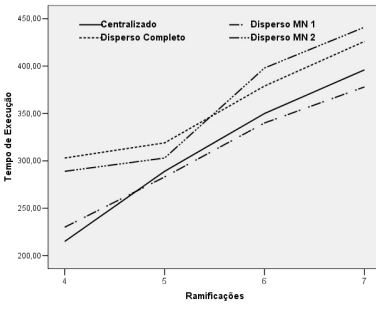


Figura 5. Impacto da variação da interdependência das atividades

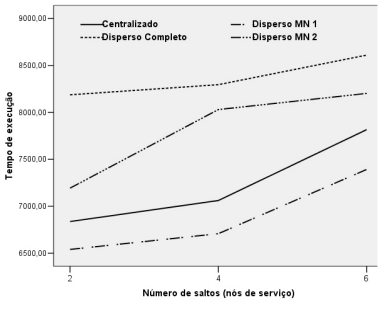


Figura 6. Impacto da variação da dispersão dos nós de serviço.

Na primeira etapa, os resultados referentes aos modelos de orquestração dispersos completo, Mn1 e Mn2 foram equivalentes. Diante disso, na Figura 4 estão representados somente os valores referentes ao modelo completo. Os particionamentos aplicados para a orquestração dispersa não variam, pois a especificação só tem duas ramificações. Nesse caso, como as atividades adicionadas representam atividades seqüenciais, para todas as variações haverá sempre no máximo duas ramificações. Conforme pode ser observado na Figura 4, a partir de cinco atividades simples por ramificação, o modelo de orquestração disperso passa a ser mais eficiente que o modelo centralizado. A melhoria associada ao modelo disperso quando o número de atividades sequenciais é aumentado pode ser atribuída ao fato de que mesmo com esse modelo encaminhando um número total de mensagens (somatório de mensagens entre nós de controle e nós de serviços e entre nós de controle) maior que o modelo centralizado, as mensagens de dados entre Nc e Ns tendem a ser enviadas por caminhos mais curtos, reduzindo, assim, o custo total de comunicação.

O gráfico apresentado na Figura 5 apresenta resultados diferentes para os modelos de orquestração dispersos completo, Mn1 e Mn2. Isso ocorre porque a especificação de orquestração utilizada nessa etapa de simulação possui um número de ramificações maior que 2, possibilitando a criação de diferentes particionamentos. Os resultados apresentados na Figura 5 mostram que o modelo de orquestração disperso Mn1 apresenta melhor desempenho a partir 5 ramificações. A explicação desse fato é similar ao da etapa anterior. É interessante notar que os resultados obtidos para os modelos disperso completo e Mn2 são sempre piores que o centralizado. Tal fato se deve a relação de vizinhança entre os nós de controle e os nós de serviços não ser vantajosa frente ao número de mensagens encaminhadas para todas as partições geradas para esses particionamentos. Isso evidencia que as partições não devem ser geradas para um conjunto pequeno de atividades (nesse caso, uma atividade), pois gera uma sobrecarga desnecessária de mensagens.

5.2 Avaliando o Impacto da Dispersão dos Nós de Serviço e de Controle

Nesse conjunto de simulações foram variados dois tipos de dispersão, um dado pelo número de saltos entre os nós de serviços e o outro pelo número de saltos entre nós de serviços e os nós de controle mais próximos a esses. Essas variações foram analisadas a luz de diferentes especificações de orquestrações.

Na primeira etapa dessas simulações, configurou-se um cenário simples usando o padrão de orquestração multimerge com 4 atividades e 4 dependências, sendo 2 atividades simples e 2 de sincronização. O número de partições geradas para todos os modelos dispersos corresponde a dois e cada partição contém uma atividade. Nesse cenário a distância entre nós de serviço (escalonados para atividades interdependentes) e a distância entre nós de serviço e de controle foi variada de 2 a 8 saltos. Os resultados de simulação mostraram que o modelo centralizado foi sempre melhor, obtendo tempos de execução cerca de 15% menor que os outros modelos, desta forma corroborando a conclusão de que esse é o modelo mais indicado para orquestrar um pequeno número de atividades. Tal resultado também evidencia que nesse tipo de cenário o comportamento das orquestrações é imune ao grau de dispersão dos nós de serviço e controle.

Visando analisar cenários mais complexos, em uma próxima etapa utilizou-se um padrão de orquestração multimerge com 34 atividades e 40 dependências, sendo 14 atividades de sincronização e 20 atividades simples. Nessa etapa, a distância entre nós de serviço foi variada de 2 a 6 saltos (Figura 6) e a distância entre nós de serviço e de controle foi variada de 1 a 3 saltos (Figura 7). A Figura 6 mostra que o modelo de orquestração disperso Mn1 é o mais eficiente quando se aumenta o grau de dispersão entre os nós de

serviços. O modelo centralizado teve pior desempenho que o Mn1, o que era intuitivamente esperado uma vez que esse cenário contém um número significativo de atividades. Contudo, o modelo centralizado obteve desempenho superior aos demais modelos dispersos, o que não era esperado. Pode-se atribuir esse resultado ao fato de que as partições geradas nos modelos Mn2 e completo contêm poucas atividades. Como o número de atividades por partição nos modelos dispersos completo e multinível Mn2 é igual a 1 no segundo passo recursivo de particionamento, o custo de comunicação acaba sendo alto nesse tipo de cenário. Ou seja, partições com número muito pequeno de atividades não justificam o particionamento.

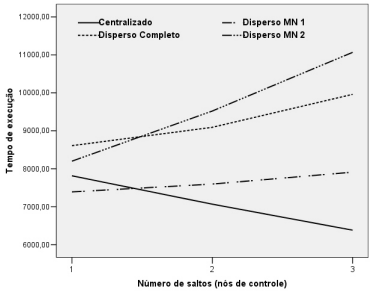


Figura 7 - Impacto da variação da dispersão nós de controle.

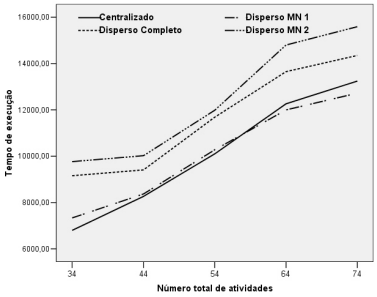


Figura 8 - Variação do número de atividades e número de saltos Nc=2.

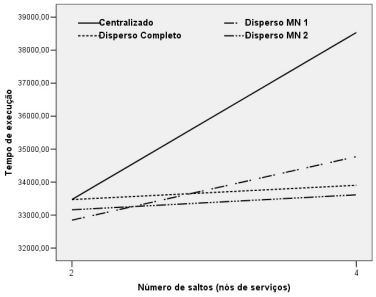


Figura 9 – Impacto com número muito grande de atividades

A Figura 7 mostra os resultados obtidos com a variação do número de saltos entre nós de serviço e de controle. O modelo centralizado apresentou um desempenho melhor quando se tem mais de um salto de distância. Isso se deve ao fato de que, como o número de mensagens encaminhadas no modelo disperso é maior que no centralizado, é necessário haver maior proximidade entre os nós de serviços e os nós de controle, dessa forma gerando caminhos mais curtos que compensem o custo adicional de comunicação. Outro resultado interessante é o melhor desempenho do modelo disperso completo frente ao modelo disperso Mn2. De forma semelhante ao que ocorre com o modelo Mn1, o modelo Mn2 necessita de um maior número de mensagens que o modelo completo. A fim de realizar uma análise mais profunda do alto desempenho observado para o modelo centralizado, em uma etapa seguinte fixou-se o número de saltos em 2 e variou-se a quantidade de atividades da orquestração. A Figura 8 mostra os resultados dessa etapa. Observa-se que a partir de cerca de 60 atividades o desempenho do modelo centralizado começa a ser inferior ao do modelo Mn1. Isso se deve ao fato de que com um grande número de atividades, a distância média dos nós de serviço em relação ao único nó de controle acaba sendo muito alta incorrendo em um custo de comunicação superior ao do modelo Mn1. Nos modelos completo e Mn2, o desempenho continuou baixo devido as partições conterem um número muito pequeno de atividades.

Na próxima etapa foi analisado um cenário com o número máximo de atividades suportado pelo simulador empregado. Configurou-se um padrão de orquestração multimerge com 166 atividades e 170 dependências, sendo 6 atividades de sincronização e 160 atividades simples. Nessa etapa, foi analisada somente a influência da dispersão dos nós de serviço entre si, variando-se a distância de 2 a 4 saltos. A Figura 9 mostra que o desempenho do modelo Mn1 é melhor que o modelo centralizado, fato que confirma a imunidade desses modelos com relação a dispersão dos nós de serviço (Figura 6). Um resultado mais interessante pode ser notado no melhor desempenho dos modelos Mn2 e completo a partir de um número de saltos igual a 3, com ligeira vantagem para o modelo Mn2. Apesar do número de mensagens de controle e de dados ser maior nos modelos completo e Mn2, com o aumento da dispersão dos nós de serviço o custo de comunicação nesses modelos acaba

sendo menor, já que as rotas percorridas por essas mensagens são otimizadas pelo posicionamento dos diversos nós de controle em relação aos nós de serviço de cada partição.

Por fim, na última etapa de simulação usou-se um cenário composto somente por nós híbridos com o intuito de avaliar o impacto da existência desse tipo de nó na rede. Os resultados obtidos demostraram que o desempenho de todos os modelos dispersos é 2% a 15% superior ao modelo centralizado nesse tipo de cenário, com vantagem para o Mn2. Esse fato já era esperado uma vez que, como nós híbridos permitem a execução local dos serviços (o mesmo nó atua como serviço e como controlador) o número de mensagens trafegadas é menor diminuindo assim o custo de comunicação da orquestração. Porém, como já mencionado, esse cenário não é realista.

Cabe ressaltar que em nenhum caso o modelo disperso completo teve desempenho melhor que os outros. Por essa razão tal modelo não aparece no algoritmo da Figura 1.

6. Trabalhos Relacionados

Em [Nanda, Chandra and Sarkar 2004] foi proposta a descentralização da gerência de execução de OWS através de heurísticas baseadas em volume de dados e número de mensagens. Neste trabalho o particionamento sempre é realizado por completo e de maneira estática. Tal abordagem não representa um cenário real para computação orientada a serviços já que, diferentemente do que se assume no Regente, considera que todos os nós de serviço contêm uma máquina de orquestração. [Nanda and Karnik 2003] trata de problemas relacionados ao sincronismo no particionamento de uma orquestração, mas não detalha como efetuar o particionamento e, tampouco, propõe soluções para o escalonamento da orquestração. Yan *et al.* (2003), Yan *et al.* (2005) e Benatallah *et al.* (2005) combinam os conceitos da tecnologia de workflow com redes P2P, apresentando uma abordagem para gerenciamento da execução de workflows de forma descentralizada. Em tais propostas as instâncias de execução são coordenadas através de comunicação direta entre os participantes do workflow através do uso da infraestrutura de redes ponto-a-ponto.

Apesar dos trabalhos mencionados proporem técnicas para execução descentralizada de orquestrações de serviços, não foram encontradas propostas de estratégias para a seleção do modelo de execução mais adequado para a gerência eficiente da execução de orquestrações de serviços Web que considerassem diferentes particionamentos, incluindo a opção de particionamento único (modelo centralizado), como é proposto no Regente.

7. Conclusões

A gerência eficiente em orquestração de SW é um dos desafios de pesquisas na área da computação orientada a serviços. Assim, novos modelos de orquestração foram propostos na literatura, nos quais as especificações são particionadas e executadas em máquinas de orquestração dispersas e independentes. Entretanto, para uma orquestração eficiente, é necessário inicialmente escolher o modelo mais adequado, levando em conta a especificação da orquestração e as características do ambiente de execução e, em seguida, escalonar de forma eficiente os planos de execução gerados, de modo a otimizar a utilização dos recursos computacionais e de rede. Com o intuito de atingir esse objetivo, o presente trabalho propôs o arcabouço Regente. As simulaçoes realizadas comprovaram a eficácia das estratégias de escolha de modelo de execução e de particionamento utilizadas pelo Regente. Um resultado interessante obtido nessas simulações é que, embora relatos da literatura indiquem o uso de abordagens descentralizadas como mais eficientes em todos os casos, comprovou-se que em cenários mais realistas (por exemplo, onde nem todos os nós são híbridos) isso nem sempre ocorre, sendo o modelo centralizado o mais adequado em alguns casos.

Referências Bibliográficas

Barr, R., Haas, Z. and Renesse, R. (2005) "Jist: An efficient approach to simulation using virtual machines", Software Practice & Experience, v. 35, p. 539-576.

Benatallah, B., Dumas, M., Sheng, Q. Z. Facilitating the rapid development and scalable orchestration of composite web services, Distributed and Parallel Databases, v. 17, n. 1, p. 5-37. January, 2005. ISSN: 0926-8782.

Binder, W., Constantinescu, I. and Faltings, B. (2006) "Service Invocation Triggers: A Lightweight Routing Infrastructure for Decentralized Workflow Orchestration", In Proc of the AINA 2006, v. 2, p. 917-921.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. (2005) "QoS-Aware Replanning of Composite Web Services", In Proc of the ICWS 05, v. 1, p. 121-129.

Chafle, G., Chandra, S., Mann, V., and Nanda, M. (2004) "Decentralized orchestration of composite web services", In Proc of the 13th WWW, ACM Press, New York, p. 134-143.

Cormen, T., et.al. (2001) "Introduction to Algorithms", MIT Press, 2nd edition, 1184 p.

Hutton, G. (2007) Programming in Haskell. Graham Hutton. Cambridge University Press.

Kim, K. and Han, D. (2001) "Performance and Scalability Analysis on Client-Server Workflow Architecture", In Proc of the ICPADS 2001, p. 179-186.

Medina, A., Matta, I., and Byers, J. (2001) " BRITE: Universal topology generation from a user's perspective", Disponível em htttp://citeseer.ist.psu.edu/medina01brite.html.

Muth, P., Wodtke, D., Weissenfels, J., Dittrich, A. K., and Weikum, G. (1998) "From Centralized Workflow Specification to Distributed Workflow Execution", In JIIS, v. 10, n. 2, p. 159-184.

Nanda, M. and Karnik, N. (2003) "Synchronization analysis for decentralizing composite Web services", In Proc of the 2003 ACM SAC, ACM Press, New York, NY, p. 407-414.

Nanda, M., Chandra, S. and Sarkar, V. (2004) "Decentralizing execution of composite web services", In Proc of the ACM OOPSLA 2004, ACM Press, New York, NY, p. 170-187.

Ouzzani, M. and Bouguettaya, A. (2004) "Efficient Access to Web Services", IEEE Internet Computing, v. 8, n. 2, p. 34-44.

Ozsu, M. and Valduriez, P. (1999) "Principles of Distributed Database Systems", Presentice Hall, 2nd Edition, 666 p., USA.

Peltz, C. (2003) "Web Services Orchestration: A Review of Emerging Technologies, Tools and Standards". Technical report, Hewlett Packard, Co.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B. and Barros A. (2003) "Workflow Patterns". Distributed and Parallel Databases, v. 14, n. 1, p. 5-51.

Woodcock et al. (1996) Using Z: Specification, Refinement, and Proof. Jim Woodcock and Jim Davies. Prentice-Hall International Series in Computer Science.

Yang, J. and Papazoglou, M. P. (2004) "Service components for managing the life-cycle of service compositions", Information Systems, v. 29, n. 2, p. 97-125.

Yan, J.; Yang, Y.; Raikundalia, G. (2003) Enacting Business Processes in a Decentralised Environment with P2P-Based Workflow Support, Proc. of WAIM 2003, Lecture Notes in Computer Science, Springer, p. 290-297, ISBN: 978-3-540-40715-7.

Yan, J.; Yang, Y.; Raikundalia, G. (2005) SwinDeW - A Peer-to-peer based Decentralised Workflow Management System, IEEE Transactions on Systems, Man and Cybernetics, v. 36, n. 5, p. 922-935. September.