

## Geração Eficiente de Planos de Materialização para Documentos XML Ativos

Daniela Pereira, Gabriela Ruberg, Marta Mattoso

Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ

{daniela, gruberg, marta}@cos.ufrj.br

**Resumo.** Um documento AXML possui dados XML representando chamadas de serviços Web. Para materializar o conteúdo de um documento AXML, é preciso invocar todas as suas chamadas de serviço Web. Gerar bons planos de materialização para documentos AXML em ambientes P2P é um problema difícil, cujo espaço de soluções cresce rapidamente. Este trabalho propõe a SLS-MC, uma estratégia de otimização baseada em busca local estocástica com múltiplas condições de parada, para a geração eficiente de planos de materialização. A SLS-MC foi implementada em um ambiente de simulação chamado SiMAX. Foram realizados vários testes que apontam o potencial de ganho de desempenho da SLS-MC.

**Palavras-chaves:** XML; serviços Web; e busca local estocástica.

**Abstract.** An AXML document consists of an XML document containing special tags that represent calls to Web services. To materialize the contents of an AXML document, all of its embedded Web service calls must be invoked. Generating good materialization plans for AXML documents in a P2P system is a hard problem whose search space is usually huge. This paper proposes an optimization strategy, called SLS-MC, which is based on stochastic local search with multiple stop conditions. A simulation tool called SiMAX was developed to enable the evaluation of different optimization strategies in complex P2P settings. The results of several experiments run with SiMAX highlight the effectiveness of the SLS-MC strategy.

**Keywords:** XML; Web services; and stochastic local search.

### 1. Introdução

A tecnologia de serviços Web se tornou uma alternativa importante para a interoperabilidade de aplicações e sistemas de integração de informações [1,5,10]. O sucesso desta tecnologia se deve principalmente à popularidade da linguagem XML e à flexibilidade de seus recursos para descrição, publicação e acesso uniforme a programas heterogêneos. Para explorar o potencial de uso de serviços Web, é necessário definir meios para combiná-los e orquestrá-los visando produzir processos mais significativos. Exemplos dessa combinação incluem os *workflows* baseados em serviços Web [3, 10,12] e os documentos **XML ativos** (ou simplesmente documentos AXML, de “*Active XML*”) [1,2,9]. Essa classe de documentos XML, surgida recentemente, possui conteúdo dinâmico, pois combina dados XML com chamadas de serviços Web. Uma grande vantagem dos documentos AXML é a possibilidade de manter atualizadas informações voláteis, como a temperatura atual de uma cidade ou um saldo bancário. Vale salientar

que um documento AXML corresponde a um *workflow* de serviços Web, no qual pode-se especificar uma chamada de serviço passando como parâmetros de entrada tanto elementos do próprio documento como resultados de outras chamadas de serviços Web do documento. A Figura 1 mostra um exemplo de documento AXML simplificado que contém uma chamada de serviço Web, representada pelo elemento “sc”. O exemplo apresenta o documento antes e depois da materialização da chamada ao serviço “*getTemperature*”. Exemplos de uso em aplicações financeiras são descritos em [12].



Figura 1. Invocação de chamada de serviço Web em um documento AXML.

As chamadas de serviço embutidas em um documento AXML consistem em **dados intencionais**, ou seja, que simbolizam os dados resultantes da execução de serviços Web. Portanto, para “**materializar**” o conteúdo completo de um documento AXML, é necessário executar todas as suas chamadas de serviço. O processo de materialização deve respeitar as dependências de execução entre as chamadas de serviço, decorrentes do aninhamento de elementos intencionais (*i.e.*, quando uma chamada de serviço recebe como parâmetros de entrada os resultados de outras chamadas). Também podem existir dependências quando um parâmetro de entrada consiste em uma consulta que seleciona outras chamadas de serviço. Essas dependências podem ser representadas em um grafo acíclico direcionado chamado de **grafo de dependências** [11].

Em ambientes distribuídos como os sistemas ponto-a-ponto (P2P) e as grades computacionais (“*Grids*”), serviços Web são geralmente providos por vários sítios. Além disso, o sítio que armazena um documento AXML (dito **sítio mestre**) pode solicitar a outros sítios a execução de chamadas de serviço contidas em sub-árvores do documento, retornando apenas os dados necessários para a construção do resultado final. Neste caso, ocorre uma **delegação** de chamadas de serviço. Um plano de materialização determina tanto o sítio que invoca como o sítio que executa cada chamada de serviço de um documento AXML, além da ordem de invocação das chamadas. Podem existir vários planos de materialização equivalentes para um documento AXML, sendo que o desempenho da materialização pode variar significativamente conforme o plano. Nesse contexto, é necessário analisar os possíveis planos e escolher aqueles com desempenho satisfatório. Todavia, como esperado em problemas de otimização em ambientes distribuídos, a materialização de documentos AXML costuma ter um espaço de busca enorme, que cresce rapidamente em relação ao número de sítios envolvidos e ao aninhamento das chamadas de serviço. A materialização de documentos AXML é um problema difícil com complexidade de tempo exponencial no pior caso [11].

Surge, então, a seguinte questão: como escolher eficientemente um bom plano de materialização, considerando cenários distribuídos dinâmicos como um sistema P2P? Nesses cenários, o tempo de otimização não pode ser longo, sob a pena de o plano escolhido não ser mais válido no momento da execução. Assim, faz-se necessário o uso de soluções heurísticas que permitam gerar seletivamente os possíveis planos. Tais heurísticas visam escolher um bom plano de materialização (*i.e.*, não necessariamente

ótimo) dentro de um tempo aceitável. Todavia, métodos de busca clássicos [13] não são aplicáveis no problema da escolha de bons planos de materialização para documentos AXML. Como o número de possibilidades é muito grande, torna-se inviável analisar e comparar todas as alternativas, como sugere a solução exaustiva. Algoritmos puramente gulosos também não são adequados, pois decisões pontuais costumam ignorar fatores importantes, como o custo de comunicação decorrente das dependências entre as chamadas de serviços. Além disso, as possibilidades de delegação em um ambiente descentralizado podem implicar em variações significativas no custo de comunicação, fazendo com que soluções parciais ótimas não garantam um ótimo global. Ou seja, planos equivalentes são comparados através de métricas de custo não monotônicas [6] e algoritmos de *Branch&Bound* também não são aplicáveis.

Uma alternativa interessante para resolver esse problema consiste no uso de algoritmos de refinamento iterativo baseados em **busca local**, cuja idéia é gerar uma solução inicial e tentar melhorá-la progressivamente analisando as possíveis mudanças de maneira incremental [13]. Para problemas muito complexos, o espaço de busca pode ser percorrido através de um **método estocástico**: a busca é iniciada com decisões aleatórias e o uso de heurísticas cresce progressivamente à medida que o conhecimento do problema aumenta. Esses algoritmos geralmente permitem encontrar boas soluções com tempos razoáveis de otimização [4,13,14].

Este trabalho contribui para a escolha de um bom plano de materialização de documentos AXML através de uma estratégia baseada em busca local estocástica com múltiplas condições de parada para a geração dos planos. A estratégia proposta é chamada de **SLS-MC** (de "*Stochastic Local Search with Multiple stop Conditions*"). Na SLS-MC, técnicas básicas de busca local são adaptadas para resolver a otimização da materialização de documentos AXML, explorando condições de parada relevantes para esse problema. Além disso, a SLS-MC permite a utilização de diferentes heurísticas de agendamento de tarefas para a geração da solução inicial.

Para avaliar a estratégia proposta, foi desenvolvido um simulador chamado **SiMAX** a partir de um protótipo do otimizador XCraft [11], que foi estendido com a estratégia SLS-MC. Vale ressaltar que o SiMAX facilita sobremaneira a avaliação experimental de diferentes estratégias de materialização de documentos AXML, permitindo a configuração de cenários com diversos tipos de documentos AXML e redes P2P arbitrariamente complexas. Visando analisar o impacto desses fatores tanto no tempo gasto na otimização como no tempo de materialização do documento, foram conduzidos vários testes com o SiMAX envolvendo diferentes condições de parada.

O restante deste artigo está dividido da seguinte maneira: a seção 2 apresenta os conceitos básicos da materialização de documentos AXML. A seção 3 descreve a SLS-MC e a seção 4 apresenta a arquitetura do SiMAX. Resultados obtidos com testes são avaliados na seção 5. A seção 6 discute os trabalhos relacionados e a seção 7 apresenta algumas conclusões e perspectivas de trabalhos futuros.

## **2. Materialização de Documentos AXML**

Para se obter uma materialização eficiente de documentos AXML, faz-se necessária a geração, busca e escolha de um plano eficiente. O primeiro passo consiste na identificação das dependências entre as chamadas de serviço [11]. Dado um documento

AXML  $D$ , o conjunto de dependências de  $D$  pode ser representado por um grafo direcionado  $G_D$ , tal que nodos denotam chamadas de serviço e, para qualquer par de nodos  $sc_i$  e  $sc_j$  em  $G_D$ , existe uma aresta do nodo  $sc_j$  para  $sc_i$  se o resultado de  $sc_j$  é usado como parâmetro de entrada (concreto ou não) de  $sc_i$ . Nossa análise considera apenas grafos válidos [11] (*i.e.*, sem ciclos de dependência). Observe que ciclos no grafo representam *deadlocks* no processo de materialização. O número de dependências de uma chamada de serviço é conhecido como “*fan-in*” e é representado por  $f_i$ .

## 2.1. Planos de Materialização Abstratos e Físicos

Um plano de materialização de um documento  $D$  consiste basicamente no conjunto de árvores geradoras (*spanning trees*) em  $G_D$ , nas quais cada nodo está associado a um operador de avaliação da respectiva chamada de serviço. Os operadores de um plano são definidos na álgebra proposta em [11]. Na estratégia SLS-MC, é considerado que as árvores obtidas de  $G_D$  são independentes entre si e podem ser analisadas e executadas individualmente. Existem dois tipos de planos de materialização: (i) **planos abstratos**, cujos operadores (do tipo “ $\mu$ ”, de *materialize*) indicam possíveis sítios provedores de serviços e possíveis delegados de execução de chamadas de serviços; e (ii) **planos físicos**, formados por operadores dos tipos *invoke* (“ $I$ ”) e *delegate* (“ $\delta$ ”), que representam a execução propriamente dita de cada chamada de serviço, tendo definido o sítio delegado e o sítio provedor. Observe que um plano abstrato denota o espaço de soluções possíveis para a materialização de um documento AXML, que pode ocorrer através de diferentes seqüências de delegação de chamadas e execução de serviços, ou seja, através de vários planos físicos equivalentes.

## 2.2. Etapas da Geração de Planos de Materialização

A Figura 2 mostra as principais etapas da geração de planos de materialização. Inicialmente, o sítio mestre de um documento AXML usa o respectivo grafo de dependências (quadro 1) e informações sobre os possíveis provedores de serviços para gerar um plano abstrato parcial (quadro 2). Cada operador do plano é anotado com um conjunto de provedores ( $\Delta e$ ). Embora o sítio mestre possa delegar a execução de cada chamada de serviço a potencialmente qualquer outro sítio da rede, consideramos que ele restringe os possíveis delegados apenas a sítios que possam executar alguma chamada de serviço contida no documento. O sítio mestre pode considerar também um conjunto conhecido de sítios que apresentem bom desempenho. Assim, são anotados no plano parcial os possíveis delegados para cada chamada de serviço, gerando um plano abstrato inicial para o documento AXML (quadro 3 na Figura 2, cujas anotações sobre provedores e delegados são representadas por  $\Delta e d$ ).

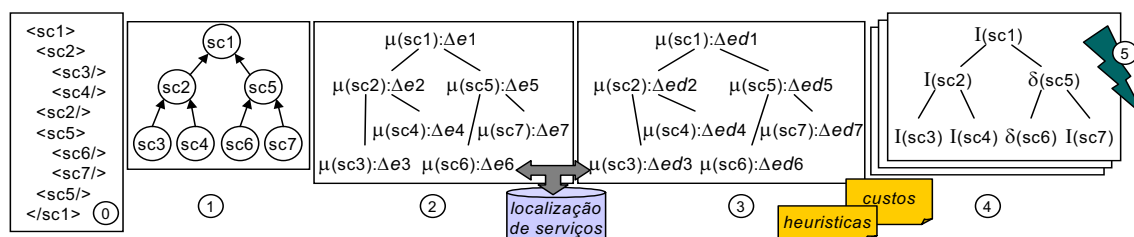


Figura 2. Principais etapas da geração de planos de materialização.

O plano abstrato inicial é usado para gerar os possíveis planos físicos (quadro 4, na Figura 2), os quais devem ser avaliados para que o melhor plano ou aqueles com desempenho satisfatório sejam identificados. Note que o espaço de busca de planos alternativos cresce rapidamente, mesmo para valores baixos atribuídos às variáveis. De fato, a busca por um plano com desempenho ótimo possui complexidade de tempo exponencial no pior caso [11] em relação ao número de planos avaliados. Para escolher o melhor plano, é necessário estimar o desempenho de cada plano gerado. Na SLS-MC, são adotadas as funções de custo definidas em [11,12], que permitem estimar o custo em **tempo de execução** de um documento AXML, ou seja, o tempo total de materialização do documento. Esse tempo é determinado principalmente pelos operadores do **caminho crítico** do plano de materialização, que é o caminho de operadores com maior custo.

Vale salientar que o grafo de dependências de um documento AXML não determina uma ordem total nas chamadas de serviço. O agendamento das invocações deve ser especificado no plano de materialização e também tem influência no desempenho. Existem várias heurísticas para o agendamento de tarefas em grades computacionais e sistemas paralelos [4,7]. Dentre as heurísticas básicas, destacam-se as seguintes: (i) **MET** (de *Minimum Execution Time*), que aloca cada tarefa à máquina com a melhor estimativa de tempo de execução, independentemente da disponibilidade da máquina; (ii) **min-min**, que identifica o menor tempo de execução de cada tarefa do plano e aloca tarefas menores primeiro, recalculando as estimativas das tarefas restantes após cada alocação; (iii) **max-min**, que é semelhante à min-min, mas que aloca tarefas maiores primeiro, sendo que a cada passo é selecionada a tarefa cujo melhor tempo de execução é o mais longo no plano; e (iv) **duplex**, que utiliza as duas heurísticas min-min e max-min para agendar as tarefas, adotando o melhor resultado encontrado.

Problemas de complexidade NP-difícil, semelhantes ao da materialização de documentos AXML, cujo objetivo é encontrar boas soluções em espaços de busca muito grandes, costumam utilizar soluções baseadas em algoritmos de busca local com condição de aceitação estocástica [3,13,14]. A questão tratada neste trabalho é a seguinte: é possível encontrar bons planos de materialização a partir do refinamento de um plano inicial, durante um período limitado, utilizando-se meios de decisão estocásticos? Este problema é atacado pela estratégia de otimização detalhada a seguir.

### **3. Busca Local Estocástica com Múltiplas Condições de Parada**

Este trabalho propõe a **SLS-MC** (de “*Stochastic Local Search with Multiple stop Conditions*”), uma estratégia baseada em busca local estocástica com múltiplas condições de parada para a geração de planos de materialização de documentos AXML. Esta solução visa selecionar um plano suficientemente bom em um tempo de otimização aceitável. Note que a SLS-MC não visa encontrar o melhor plano, mas evitar os piores.

A Figura 3 mostra a estratégia adotada na SLS-MC. Um plano abstrato inicial anotado com informações sobre os provedores e os possíveis delegados das chamadas de serviços é fornecido como entrada. A partir do plano abstrato, é gerada uma solução inicial (ou seja, um plano físico) usando uma heurística de agendamento de tarefas. Então, é estimado o custo da solução inicial e identificado o seu caminho crítico. A estratégia consiste em iterações nas quais são geradas modificações nos operadores que compõem o caminho crítico do plano, até que as condições de parada sejam satisfeitas. A cada iteração, o custo da solução obtida é avaliado. Se o custo for melhor que o da



Durante a busca, é guardado o melhor plano encontrado em todas as iterações, que é atualizado sempre que um plano com custo inferior é gerado. Após a última iteração, o melhor plano é utilizado para a materialização do documento. A busca termina quando as condições de parada são alcançadas. A SLS-MC considera que múltiplas condições de parada podem ser especificadas em uma expressão lógica que é verificada a cada iteração. Por exemplo, pode-se determinar que a busca pare se for encontrado um plano com custo inferior a 50% da solução inicial *ou* se cem mil planos forem gerados. Neste caso, temos duas condições de parada e a busca será interrompida quando qualquer uma dessas condições for satisfeita. Basicamente, as condições de parada estão relacionadas a: **ganho de custo** (em relação à solução inicial), e **tempo gasto na otimização**. Além disso, se o número máximo de reinícios permitidos for alcançado, indicando que a melhor solução já foi encontrada, a busca é finalizada. Para garantir que a busca termine, caso nenhuma condição de parada seja atingida, é utilizada também uma **condição de teto** (e.g., número máximo de planos gerados). Observe que o teto deve representar uma condição que dure o suficiente para não influenciar a busca.

#### **4. SiMAX – Um Simulador para a Materialização de Documentos AXML**

A complexidade e o dinamismo dos sistemas P2P introduzem várias dificuldades na realização de testes, pois é preciso dispor de um grande número de máquinas, distribuídas e conectadas entre si, cada uma com configurações próprias. Isto muitas vezes não é viável, ou torna-se muito difícil ter acesso direto para configurar as máquinas com os parâmetros desejados. Devido a essas dificuldades, este trabalho propõe uma ferramenta de simulação chamada **SiMAX** para facilitar a realização de testes na avaliação da estratégia SLS-MC. O simulador SiMAX foi desenvolvido com a linguagem de programação Java e utiliza o ActiveXML [1], uma plataforma P2P de *software* livre, como substrato para gerência de documentos AXML. O SiMAX estende o otimizador XCraft [11] com módulos que implementam a SLS-MC.

##### **4.1. Arquitetura do SiMAX**

A Figura 4 apresenta uma visão geral da arquitetura do SiMAX. Os componentes do XCraft que foram estendidos pelo SiMAX para apoiar a estratégia SLS-MC são representados com linhas duplas. No SiMAX, as configurações necessárias para a realização de uma simulação são passadas através de arquivos XML. O XCraft foi totalmente encapsulado pelo simulador, tal que as entradas e saídas referentes à materialização de um documento AXML foram direcionadas para módulos do SiMAX.

O SiMAX funciona carregando inicialmente as configurações necessárias para a simulação a partir de arquivos XML. Estas configurações são utilizadas para popular os catálogos de informações e estatísticas do sítio ActiveXML. Além disso, as configurações também especificam o **perfil de otimização** do XCraft, que consiste em um conjunto de propriedades que determinam o comportamento da estratégia de otimização utilizada na simulação. Um arquivo XML específico descreve o perfil de otimização. Contudo, a maior parte dos arquivos XML contém informações sobre os sítios do sistema P2P, os *links* de comunicações entre eles, os serviços Web disponíveis e o documento AXML que deve ser materializado.

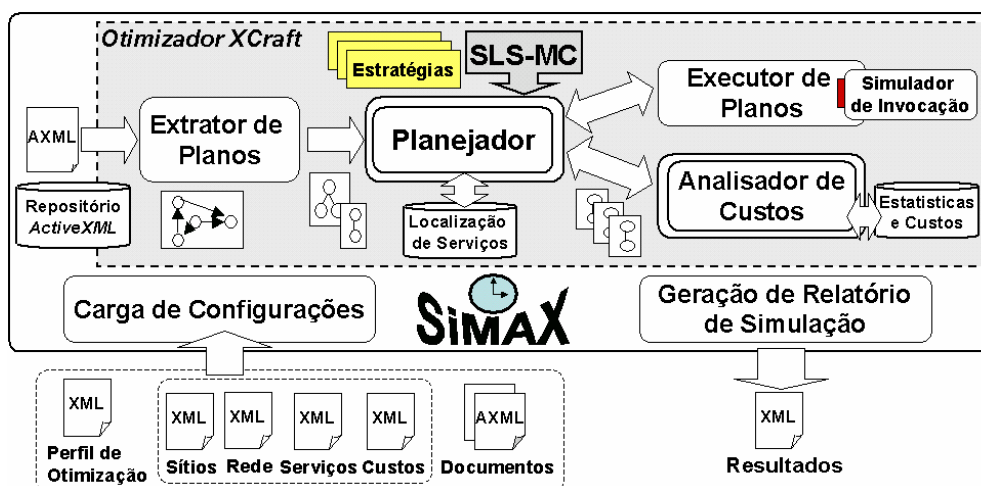


Figura 4. Arquitetura do SiMAX.

Feita a carga inicial das configurações do ambiente de simulação, o SiMAX está pronto para disparar a análise da materialização de um documento AXML. Basicamente, o documento AXML é passado para o otimizador XCraft, que extrai o grafo de dependências e gera um plano abstrato inicial. Esse plano é encaminhado ao Planejador do XCraft, que utiliza informações sobre a localização dos serviços para anotar nos operadores do plano os respectivos sítios provedores e possíveis delegados. Conforme a estratégia de otimização configurada, o Planejador do XCraft usa o plano abstrato inicial para gerar e analisar planos físicos candidatos, solicitando ao Analisador de Custos para estimar o tempo de execução de cada plano gerado. O SiMAX estendeu o componente Planejador do XCraft com os algoritmos de agendamento de tarefas MET, min-min, max-min e duplex. Também foi acrescentado um componente que implementa todas as etapas da estratégia SLS-MC.

No Executor de Planos, onde cada plano selecionado é materializado, o SiMAX incorporou um mecanismo para a simulação da invocação de chamadas de serviços. Monitores de desempenho atuam em todo o sistema e relatórios contendo os resultados obtidos em uma simulação, tais como tempo de otimização e tempo de execução dos planos de materialização, são armazenados em arquivos XML para análise posterior. Um relatório inclui também os planos escolhidos. É importante ressaltar que apenas a execução dos planos de materialização escolhidos é simulada no SiMAX. Todo o processo de otimização é devidamente executado, permitindo assim uma avaliação bem realista das estratégias simuladas.

#### 4.2. Configuração do Ambiente de Simulação

O SiMAX permite definir um conjunto de propriedades que descrevem os elementos envolvidos no ambiente simulado. Estas propriedades são divididas em duas categorias: perfil de otimização; e estatísticas e custos. O perfil de otimização determina a estratégia a ser utilizada pelo otimizador, bem como todos os parâmetros necessários para a execução da estratégia escolhida. Na Tabela 1 são descritas algumas propriedades do perfil de otimização do SiMAX. Observe que um documento tanto pode ser materializado sem nenhuma otimização, como também pode ser otimizado sem que a materialização seja executada (*i.e.*, somente é feita a geração do plano). A propriedade *Search* pode ser configurada como sendo *Exhaustive*, para busca exaustiva,

ou SLS, para busca local. Condições de paradas são determinadas pelas propriedades *CostFactor* e *PlansFactor*. Já a propriedade *PlansCeiling* representa a condição de teto utilizada se a estratégia SLS-MC for selecionada.

**Tabela 1 – Propriedades do perfil de otimização do SiMAX.**

<i>Optimize</i>	Indica se o simulador deve otimizar o documento a ser materializado.
<i>Delegate</i>	Caso haja otimização, determina se a delegação pode ser utilizada.
<i>Materialize</i>	Determina se o simulador deve materializar o documento.
<i>Blocking</i>	Indica se operadores de um plano podem ser executados paralelamente (uso de <i>threads</i> ).
<i>Search</i>	Determina método de busca utilizado.
<i>Heuristic</i>	Indica a heurística de agendamento de tarefas a ser utilizada.
<i>CostFactor</i>	% de ganho de custo relativo ao custo da solução inicial na condição de parada.
<i>PlansFactor</i>	% de planos gerados relativo à complexidade do documento para parada.
<i>PlansCeiling</i>	Número máximo de planos gerados.
<i>HighCostProbability</i>	Probabilidade de aceitação de planos piores para a solução estocástica ( <i>fpa</i> ).
<i>StabilizationFactor</i>	% de custo que indica se uma solução estabilizou ou não.
<i>NumPlansStabilize</i>	Número de planos semelhantes para que a solução seja considerada estabilizada.
<i>NumRestarts</i>	Número máximo de reinícios permitidos.

**Tabela 2 - Propriedades de estatísticas e custos do SiMAX.**

<i>InitClientTime</i>	Tempo necessário para inicializar os módulos de software no cliente.
<i>InitServerTime</i>	Tempo necessário para inicializar os módulos de software no cliente.
<i>Pack</i>	Tempo médio de empacotamento de 1 byte pelo SOAP.
<i>Unpack</i>	Tempo médio de desempacotamento de 1 byte pelo SOAP.
<i>Parse</i>	Tempo médio para compilar um fragmento AXML.
<i>PlanAnalyseTime</i>	Tempo médio gasto na análise de um plano de materialização.
<i>Processing</i>	Fator de carga de processamento de um sítio.

As propriedades de estatísticas e custos do SiMAX descrevem as características físicas do ambiente simulado. Elas são definidas para cada sítio, especificando os custos de comunicação de dados e de execução dos serviços Web. Essas propriedades são essenciais para as funções que estimam os custos dos planos de materialização. As principais propriedades de estatísticas e custos são apresentadas na Tabela 2. Existem também as propriedades sobre os sítios do sistema P2P, a distribuição dos serviços Web utilizados e o documento AXML avaliado.

## 5. Avaliação da Estratégia Proposta

Para analisar os vários fatores que possuem impacto na estratégia proposta, foram realizados com o SiMAX diversos testes comparativos. O SiMAX foi executado em uma máquina AMD Athlon de 1.8GHz com 512 *Mbytes* de memória RAM e sistema operacional *Windows*. Estes testes procuraram avaliar o desempenho da SLS-MC através da materialização de diferentes tipos de documentos AXML, usando várias heurísticas de agendamento de tarefas e condições de parada.

Foram gerados três tipos de documentos AXML: com 33, 65 e 129 chamadas de serviço, representados por **d33**, **d65** e **d129** respectivamente. Esses documentos usam 15 serviços Web distintos e possuem chamadas de serviço aninhadas com altura fixa em  $h=3$ , permitindo assim que haja comunicação de dados e possibilidade de delegação de tarefas. Cada documento possui uma chamada de serviço que é o nodo raiz, cujos filhos também são chamadas de serviço e têm  $f_i=3$ , ou seja, três chamadas de serviço como parâmetros de entrada. Esses documentos possuem espaços de busca grandes mesmo em cenários com poucos sítios, permitindo avaliar o uso efetivo da estratégia SLS-MC.

Serviços Web são representados na simulação por diferentes tempos de execução, sendo providos por quatro sítios distintos. Esses sítios estão conectados através de diferentes *links* de comunicação, conforme a Figura 5. Os números sobre as conexões indicam uma relação entre elas (*e.g.*, o *link* entre os sítios 1 e 3 é 40 vezes mais rápido que entre os sítios 1 e 4). O sítio mestre é o Sítio 4. O número de cada sítio indica também capacidade computacional; ou seja, quantas vezes cada sítio é mais lento em relação ao sítio 1. Por exemplo, o Sítio 4, é quatro vezes mais lento que o Sítio 1. Considera-se um cenário onde todos os sítios podem executar todos os serviços Web e todas as chamadas de serviço podem ser delegadas a qualquer sítio.

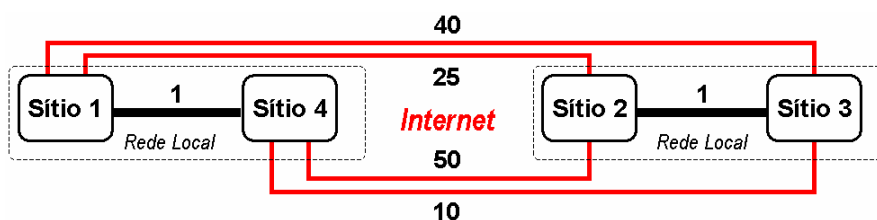


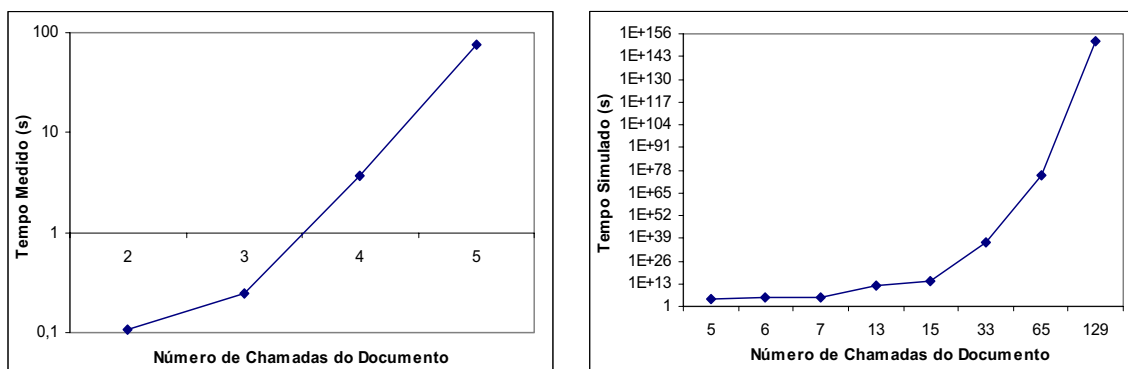
Figura 5. Topologia P2P utilizada nos testes.

O tempo de execução de cada chamada de serviço foi calculado considerando-se o serviço Web requisitado, o sítio envolvido e o tamanho do resultado (pressupondo, por simplicidade, resultados com um tamanho fixo de 100Kbytes para cada chamada). Os serviços Web possuem tempos de processamento distintos. Em cada sítio, os valores relativos aos tempos de empacotamento e desempacotamento de mensagens, além de outras características ligadas às propriedades de estatísticas e custos, foram configuradas em função da capacidade do sítio. Ou seja, o tempo de empacotamento no Sítio 4 é quatro vezes mais lento que o tempo levado pelo Sítio 1.

Foram conduzidas três baterias de testes com as seguintes estratégias: (i) estratégia exaustiva; (ii) SLS-MC com parada por número máximo de planos avaliados; e (iii) SLS-MC com parada por ganho mínimo de custo relativo à solução inicial. Nas baterias com a estratégia SLS-MC, variou-se a heurística de agendamento de tarefas para cada documento. Nas baterias (ii) e (iii), foi analisada também a estratégia gulosa equivalente a cada heurística de agendamento de tarefas, para se estabelecer um referencial de custo de execução dos planos. Foi utilizado um teto de 500.000 planos avaliados. Cada teste foi executado 4 vezes e utilizou-se a média dos resultados obtidos.

### 5.1. Análise da Estratégia Exaustiva

Nesta bateria, analisou-se a estratégia exaustiva com o objetivo de determinar o tempo máximo de otimização dos planos. Todavia, verificou-se que a complexidade da busca (*i.e.*, o número de planos físicos equivalentes possíveis) nos documentos avaliados é muito grande, tornando impossível a sua execução prática. Assim, foram gerados documentos pequenos e medido o tempo de otimização da solução exaustiva para esses documentos, cujos resultados estão na Figura 6(a). Para os documentos **d33**, **d65** e **d129**, foi estimado o tempo de otimização a partir da respectiva complexidade, considerando-se um tempo médio de geração e avaliação de cada plano (estimado em 0.5 ms para o Sítio 1). Os resultados dessa simulação são mostrados na Figura 6(b).

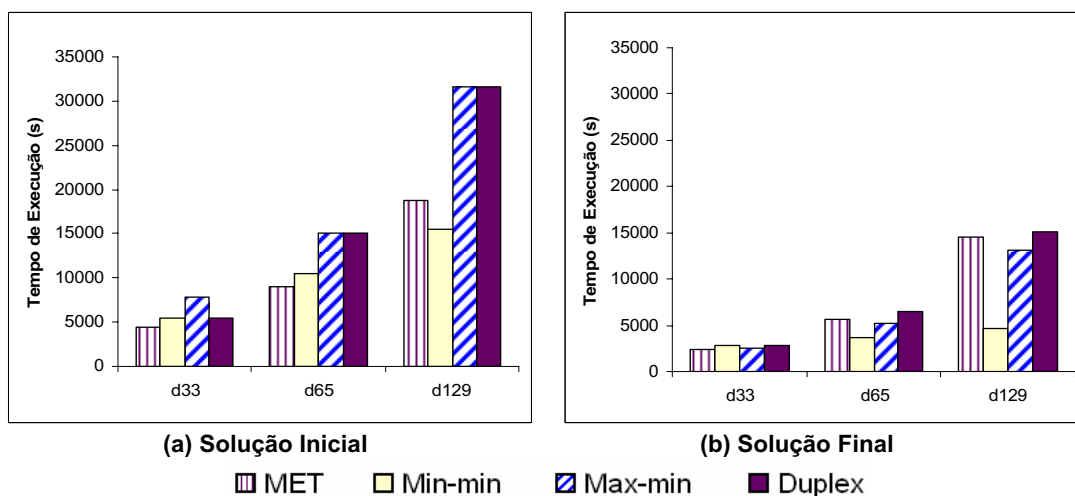


(a) (b)  
Figura 6. Tempo de otimização da estratégia exaustiva.

Observe que o tempo necessário para a execução de uma estratégia exaustiva é impraticável mesmo para o **d33**. Por ser uma abordagem mais escalável, a estratégia SLS-MC permite determinar o tempo de otimização conforme as condições de parada utilizadas, podendo-se fixar um limite. A estratégia SLS-MC permite que o desempenho do plano seja melhorado sem exigir a completa varredura do espaço de busca. Vale salientar que isso é bem adequado quando se trata de serviços Web. Na especificação do SLA (*Service Level Agreement*) [5] de um serviço Web, é comum o responsável pelo serviço determinar o tempo máximo de execução, sendo desejável que ele consiga também especificar o quanto deste tempo pode ser gasto em otimização.

## 5.2. Análise da Qualidade do Plano conforme a Heurística de Agendamento

Os resultados dessa bateria de testes são mostrados na Figura 7. Esses testes visam analisar o impacto das heurísticas de agendamento de tarefas na qualidade dos planos obtidos pela estratégia de otimização. A condição de parada é fixada como número de planos avaliados, calculado em 5% da complexidade (*i.e.*, do número total de planos possíveis). Na Figura 7(a), é mostrado o custo do plano correspondente à solução inicial, obtida de forma gulosa, para cada heurística de agendamento de tarefas. Na Figura 7(b) temos os custos finais dos planos obtidos pela SLS-MC a partir das soluções iniciais.



(a) Solução Inicial (b) Solução Final  
Figura 7. Resultados de diferentes heurísticas de agendamento de invocações.

Comparando os custos iniciais e finais obtidos por cada heurística, a SLS-MC consegue reduzir em média 50% os tempos de execução gerados inicialmente de forma gulosa, independente da heurística utilizada. A heurística min-min apresentou maior ganho: cerca de 64% em média para os documentos avaliados. Já a heurística MET obteve 35% de ganho. Embora abaixo dos resultados da min-min, esse valor ainda é interessante em relação à solução inicial gulosa.

### 5.3. Análise de Impacto da Condição de Parada

O objetivo desta bateria de testes é verificar o desempenho da SLS-MC conforme a condição de parada da estratégia. A idéia é analisar o tempo que cada heurística requer para alcançar resultados satisfatórios (ou estabilizar). As condições de parada avaliadas foram: número de planos avaliados (Condição 1), estabelecido em 5% da complexidade do documento; e ganho de custo (Condição 2), fixado em 30% do custo da solução inicial. A Figura 8 mostra os tempos de otimização obtidos.

Na Condição 1 (Figura 8(a)), a estratégia SLS-MC min-min estabilizou antes de atingir os 5% da complexidade total, sendo mais rápida que as demais heurísticas para todos os documentos. Nas demais heurísticas, principalmente com documentos maiores, a busca foi interrompida pela condição de teto antes de atingir a condição de parada. Na Condição 2, para documentos maiores a estratégia SLS-MC min-min também foi a mais rápida. Para documentos menores, no entanto, acabou sendo a mais lenta. Este resultado provavelmente se deve ao fato de a solução inicial gerada pela min-min ser muito próxima da solução final. Como a Condição 2 pára conforme o ganho em relação à solução inicial, a busca perde muito tempo tentando melhorar uma solução que já é suficientemente boa. A heurística min-min tende a reduzir o tempo de otimização para atingir um determinado ganho de custo à medida que o tamanho do documento aumenta.

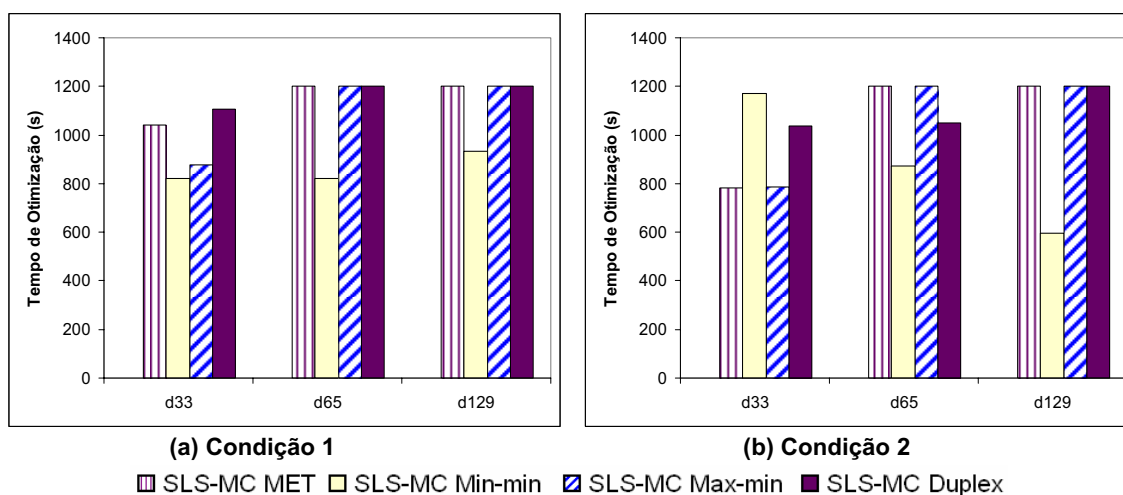


Figura 8. Tempo de otimização de diferentes condições de parada.

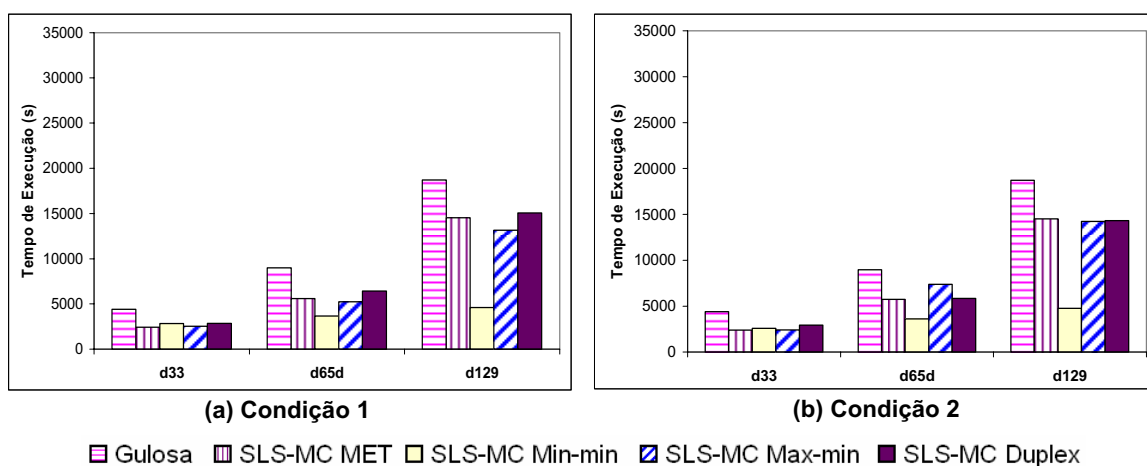


Figura 9. Tempos de execução obtidos em diferentes condições de parada.

Os custos dos planos obtidos em cada heurística são exibidos na Figura 9. Na impossibilidade de determinar o custo do plano ótimo (pois a estratégia exaustiva é impraticável), os tempos de execução obtidos com a SLS-MC são comparados com os tempos de execução de uma estratégia gulosa baseada na heurística MET. Em todos os cenários avaliados, verificou-se que a estratégia SLS-MC permite obter ganho de desempenho em um tempo razoável de otimização. Esse ganho pode ainda ser potencializado com o uso de heurísticas nas modificações do caminho crítico, de acordo com os serviços Web acessados e as configurações P2P do sistema.

## 6. Trabalhos Relacionados

As aplicações distribuídas como os *workflows* baseados em serviços Web e documentos AXML têm em comum o fato de atuarem em ambientes extremamente dinâmicos, regidos por um grande número de variáveis que determinam a configuração e o comportamento destas aplicações. Todo este dinamismo introduz várias dificuldades na escolha do plano de execução do *workflow* ou do plano de materialização do documento AXML. Este último acrescenta dificuldades ao contemplar grafos de dependências complexos, a possibilidade de delegação de tarefas em ambientes P2P e um conjunto específico de provedores para cada chamada de serviço Web.

A materialização de documentos AXML foi inicialmente abordada em [2,9]. Milo *et al.* [9] propõem um método para garantir a execução correta da materialização quanto à validade do tipo do documento resultante. Já em [2], é considerada a replicação de serviços Web e é proposta uma estratégia baseada em custo para escolher a melhor réplica. No entanto, não é considerada a delegação de execuções e presume-se uma busca exaustiva. Recentemente, foi proposto o otimizador XCraft [11], cuja estratégia divide o espaço de busca em partes menores visando reduzir a complexidade total do problema e permitir uma melhor adaptação ao dinamismo dos ambientes P2P. A idéia geral do XCraft é alternar planejamento e execução, quebrando um plano abstrato inicial em tarefas e essas tarefas em subplanos de altura “ $k$ ” (parâmetro a ser determinado na inicialização da estratégia). Para cada subplano abstrato, são gerados e analisados exaustivamente planos físicos equivalentes, tal que o subplano escolhido é executado antes de prosseguir com a otimização. Porém, uma dificuldade encontrada é a escolha do parâmetro “ $k$ ”: em função das dependências entre as chamadas de serviço de

um documento AXML, até mesmo valores pequenos de “ $k$ ” podem envolver espaços de busca muito grandes. Esse problema motivou o desenvolvimento da estratégia SLS-MC, que é complementar à proposta do XCraft. Vale ressaltar, contudo, que a SLS-MC não é restrita ao contexto do XCraft, podendo ser aplicada na otimização de vários problemas relacionados à distribuição de *workflows* baseados em serviços Web.

A geração de planos de materialização para documentos AXML pode ser comparada à otimização de consultas distribuídas envolvendo multi-junções, com a complexidade adicional do aninhamento de chamadas. Segundo [8], quando o número de relações é maior que 5 ou 6 e é necessário realizar multi-junções, a abordagem exaustiva se torna muito onerosa e os autores mostram experimentalmente a eficiência de estratégias heurísticas com uso de perturbação aleatória. Em [8] é proposto um algoritmo para ordenação eficiente de planos de consultas sobre um esquema mediado. Para documentos AXML, ao contrário do que ocorre em [8], não é possível aplicar uma função de custo monotônica, devido ao aninhamento das chamadas de serviço e à possibilidade de delegação das execuções.

No contexto do processamento de *workflows* em sistemas paralelos, existem vários algoritmos para o agendamento de tarefas de um grafo (ou seja, *list scheduling* e suas variações [4,7]). Tais algoritmos constituem a base das estratégias para distribuição de tarefas de *workflows* em *Grids*. Contudo, essas estratégias são geralmente gulosas e consideram que as tarefas podem ser arbitrariamente atribuídas às máquinas, o que não ocorre com as chamadas de serviços de um documento AXML. Em [3], é proposto um método baseado em busca local para agendar a execução de *workflows* considerando todas as tarefas envolvidas. Esse método permite aceitar alternativas de desempenho inferior durante a geração do espaço de busca, visando escapar de mínimos locais. Já em [14], o agendamento de *workflows* é baseado no algoritmo TASK, que explora a topologia dos nodos de um *workflow* para melhorar o desempenho da busca local. No TASK, o refinamento do plano é direcionado por mudanças seletivas que atingem principalmente nodos com alto custo. Essa idéia também é explorada no SLS-MC. Porém, tanto [14] como [3] não consideram a delegação de partes do problema, além de pressuporem um *pool* de máquinas dedicadas para o agendamento de tarefas.

## **7. Conclusão e Trabalhos Futuros**

A principal contribuição deste trabalho consiste na estratégia SLS-MC, uma solução baseada em busca local estocástica com múltiplas condições de parada que visa minimizar o tempo de otimização da materialização de documentos AXML. Vale ressaltar que esta abordagem é inovadora tanto no contexto dos documentos AXML como para *workflows* baseados em serviços Web. Ao contrário do que ocorre nas demais estratégias baseadas em busca local para *workflows*, a SLS-MC considera que sítios podem colaborar através da delegação de tarefas e que cada tarefa do *workflow* possui um conjunto específico de provedores.

As técnicas propostas foram implementadas em uma ferramenta de simulação chamada SiMAX, que foi utilizada na realização de diversos testes com diferentes configurações de redes, máquinas, documentos AXML e estratégias de geração de planos de materialização. Os resultados obtidos indicam que a solução proposta apresenta um grande potencial de ganho de desempenho em relação aos métodos exaustivo e guloso. Além disso, esse potencial se confirma para diferentes técnicas de

agendamento de tarefas, principalmente para a heurística min-min. Nos testes realizados, a min-min apresentou um melhor desempenho, produzindo não só resultados mais rapidamente que as demais como também com custo menor, principalmente para documentos maiores (onde a otimização é essencial). Uma perspectiva para evoluir a proposta do SLS-MC consiste na especificação de uma versão distribuída da estratégia, tal que o espaço de busca seja dividido e distribuído entre sítios que colaboram entre si para resolver o problema.

**Agradecimentos.** Este trabalho é financiado parcialmente pelo CNPq. Gabriela Ruberg agradece também ao Banco Central do Brasil pelo apoio financeiro. O conteúdo deste artigo expressa exclusivamente a opinião das autoras.

### **Referências Bibliográficas**

1. S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo e R. Weber: Active XML: Peer-to-Peer Data and Web Services Integration. VLDB 2002, pp. 1087-1090.
2. S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu e T. Milo: Dynamic XML documents with distribution and replication. SIGMOD Conference 2003, pp. 527-538.
3. J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal e K. Kennedy: Task Scheduling Strategies for Workflow-based Applications in Grids, CCGrid 2005.
4. T. Braun, H. Siegel e N. Beck: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, 2001, v. 61(6), pp. 810-837.
5. H. Haas: Web Services Activity at W3C. Disponível em <http://www.w3.org/2002/ws>
6. A. Doan e A.Y. Halevy: Efficiently Ordering Query Plans for Data Integration. ICDE 2002, pp.393-402.
7. Y.-K. Kwok e I. Ahmad: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. 31(4): 406-471 (1999)
8. R. S. G. Lanzelotte, P. Valduriez e M. Zaït: On the Effectiveness of Optimization Search Strategies for Parallel Execution Spaces. VLDB 1993, pp. 493-504.
9. T. Milo, S. Abiteboul, B. Amann, O. Benjelloun e F. Dang-Ngoc. Exchanging Intensional XML Data. SIGMOD Conference 2003, pp. 289-300.
10. P. Pires, M. Mattoso e M. Benevides: Mediating Heterogeneous Web Services. SAINT 2003, pp. 344-347.
11. G. Ruberg e M. Matoso: XCraft: A Dynamic Optimizer for the Materialization of Active XML Documents. Relatório Técnico, COPPE/UFRJ, 2006.
12. N. Ruberg, G. Ruberg e I. Manolescu: Towards Cost-based Optimization for Data-intensive Web Service Computations. SBBD 2004, pp. 283-297.
13. S. Russel e P. Norvig. Inteligência Artificial, Editora Campus, 2ª. Edição, pp. 1040, 2003.
14. M.-Y. Wu, W. Shu e J. Gu: Efficient Local Search for DAG Scheduling. IEEE Trans. Parallel Distrib. Syst. 12(6): 617-627 (2001).