Uma infra-estrutura para execução dinâmica de serviços em grades computacionais

Carlos R. Senna, Luiz F. Bittencourt e Edmundo R. M. Madeira

Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6.196 – 13.083-970 – Campinas – SP – Brasil

{crsenna, bit, edmundo}@ic.unicamp.br

Abstract. *Computational grids based on OGSA and WSRF offer suitable support for heterogeneity of resources and diversity of applications. However, they do not totally support the dynamic nature of these environments. In this paper we present an infrastructure for workflow execution with dynamic service deployment support for grids. Our infrastructure, mainly composed of a service scheduler and a workflow engine, joins dynamic instantiation of services to on-demand provisioning in workflows execution, reaching better scalability, flexibility, and availability of resources as well as it improves the environment robustness. The paper describes the proposed infrastructure architecture and shows some results from an implementation developed for the Globus Toolkit 4.*

Resumo. *As grades computacionais baseadas em OGSA e WSRF oferecem suporte adequado para a heterogeneidade de recursos e a diversidade de aplicações. No entanto tem sérias limitações para suportar a dinâmica natural desses ambientes. Neste artigo apresentamos uma infra-estrutura para a execução de workflows com suporte a publicação dinâmica de serviços. Nossa infra-estrutura, formada em seu núcleo por um escalonador de serviços e um gerente de workflows, agrega instanciação dinâmica de serviços com provisionamento sob demanda durante a execução dos workflows melhorando a escalabilidade, a flexibilidade, a disponibilidade de recursos e a robustez das grades. No artigo descrevemos a arquitetura da infra-estrutura proposta e mostramos alguns resultados da implementação feita para o Globus Toolkit 4.*

1. Introdução

As grades são ambientes computacionais interligados através de uma rede, local ou remota, na qual recursos heterogêneos podem ser compartilhados [Foster 2002, UFCG 2008, Goldchleger 2003]. As grades permitem que instituições e pessoas possam compartilhar recursos e objetivos, através de regras de segurança e políticas de utilização, formando as Organizações Virtuais [Foster 2002]. O padrão *Open Grid Services Architecture* (OGSA) [Foster 2002] propõe que a interoperabilidade dos recursos heterogêneos da grade seja feita através dos protocolos da Internet, permitindo às grades usarem os padrões e os paradigmas da computação orientada a serviços (*Service-Oriented Computing* - SOC) [Curbera 2003]. A especificação *Web Services Resource Framework* (WSRF) [OASIS WSRF 2006], que une iniciativas das comunidades de grades e de *Web Services*, provê flexibilidade, extensibilidade e manutenção de estado para as grades. No entanto as implementações atuais, como o

Globus Toolkit 4 [Globus 2008], ainda apresentam sérias limitações para suportar a dinâmica natural das grades.

O ambiente naturalmente colaborativo das grades permite aos usuários estabelecerem ligações entre os serviços, organizando-os na forma de *workflows* ao invés de construírem aplicações tradicionais através das linguagens de programação. Tais composições de serviços trazem novas exigências devido à interação entre os participantes nas organizações virtuais, notadamente no que se refere à coordenação e composição dinâmica de processos e serviços.

Um exemplo dessas novas exigências pode ser visto quando um recurso computacional falha durante a execução de um *workflow* de longa duração. A recuperação é possível se a infra-estrutura permitir o redirecionamento para outro recurso. Para que o redirecionamento possa ser feito com sucesso outros fatores são necessários, como encontrar um novo recurso que atenda aos requisitos e tenha os serviços prontos para o uso e redirecionar as referências com os demais recursos participantes do *workflow*. O novo recurso entra na composição recebendo como entrada a saída dos antecessores do recurso substituído e direciona a sua saída para o sucessor do recurso substituído. Um outro exemplo pode ser o aproveitamento eficiente dos recursos que entram e saem em grades oportunistas. Não é suficiente que a infra-estrutura perceba o novo recurso à disposição. O recurso deve ser adequado estando com os serviços disponíveis para permitir a execução de qualquer *workflow*. Para garantir o uso eficiente dos recursos itinerantes seria necessário que todos os recursos tenham todos os serviços instalados e prontos para o uso, ou que a infra-estrutura tenha a capacidade de provê-los sob demanda.

Neste trabalho apresentamos uma infra-estrutura que agrega provisionamento sob demanda na execução de *workflows* em grades computacionais. Nossa infra-estrutura, formada em seu núcleo por um escalonador de serviços e um gerente de *workflows*, permite instanciação dinâmica de serviços, com provisionamento sob demanda, durante a execução dos *workflows* melhorando a escalabilidade, a flexibilidade, a disponibilidade de recursos e a robustez das grades. A implementação feita agrega a facilidade de execução dinâmica de serviços em *workflows* ao GT4, uma vez que este não oferece tal recurso. Usamos também a implementação em experimentos a fim de avaliarmos os custos envolvidos com a publicação dinâmica de serviços.

O artigo descreve alguns conceitos e trabalhos relacionados à proposta na Seção 2; mostra a infra-estrutura para a execução dinâmica de serviços na Seção 3; descreve a arquitetura proposta na Seção 4; comenta os aspectos da implementação feita mostrando alguns resultados obtidos na Seção 5; e na Seção 6 mostra a conclusão e os trabalhos futuros.

2. Conceitos e Trabalhos Relacionados

Nossa infra-estrutura é direcionada a grades que usam o paradigma da computação orientada a serviços. Nesse contexto alguns conceitos e padrões são importantes e convém destacá-los.

O GT4 é uma implementação da OGSA, proposta pelo *Open Grid Forum* (OGF), onde os recursos lógicos e físicos são modelados como serviços, permitindo

dessa forma que a grade use os conceitos da computação orientada a serviços. Um serviço é composto de uma interface definida através do padrão *Simple Object Access Protocol* (SOAP), descrito em *Web Services Description Language* (WSDL) e implementado por métodos, configurações e suporte de bibliotecas. Serviços podem também ser composições de outros serviços. O GT em sua versão 4 ampliou o suporte aos serviços incorporando as especificações WSRF.

WSRF é um grupo de especificações que modela o acesso a recursos que podem manter o seu estado usando *Web Services*. Um *WS-Resource* é a composição de recurso e *Web Service* através do qual o recurso pode ser acessado. Cada *WS-Resource* tem uma identificação única (*endpoint reference*) composta pela *Uniform Resource Identifier* (URI) do serviço e a chave de acesso ao estado do recurso.

O GT4 provê um *container* para abrigar os *WS-Resources* da grade e todo recurso computacional deve ter um *container* ativo. Um *container* é um sistema baseado em *Web Services* que hospeda serviços e atende a requisições de clientes que invocam operações definidas no serviço. Antes de serem usados em aplicações ou composições, os serviços devem ser publicados junto aos *containers* existentes em cada um dos recursos computacionais participantes da grade.

A dinâmica das grades e das organizações virtuais reforçam a necessidade de provisionamento sob demanda, onde os requisitos organizacionais devem nortear a configuração do sistema. Cabe ao ambiente prover dinamicamente os serviços relacionados com cada aplicação no momento do uso, não sendo recomendável disponibilizar todos os serviços em todos os recursos computacionais da grade.

Para permitir o provisionamento sob demanda é necessário suportar a instanciação dinâmica de serviços, isto é, colocar o serviço no recurso computacional, publicá-lo para que possa ser tratado por um *container* e ativar o *container* para que este possa atender as requisições ao serviço.

Alguns trabalhos propõem soluções para suportar publicação dinâmica de serviços em grades. [Weissman 2005] implementa uma arquitetura para publicação de serviços baseada no Tomcat para estender o GT3. Essa implementação adiciona ou substitui serviços sem interromper o *container*, mas por usar o Tomcat não oferece o desempenho adequado. [Sun 2005] mostra uma solução para publicação remota para o CROW Grid, mas não mostra os detalhes sobre sua capacidade e disponibilidade e [Watson 2005] foca o problema de publicação dinâmica de serviços, mas não é orientado a WSRF.

DynaGrid [Byun 2007] propõe um arcabouço para construção de grades para aplicações compatíveis com WSRF. DynaGrid provê mecanismos para publicação dinâmica, migração de recursos e acesso a instâncias de modo transparente. No entanto não descreve a solução para o reinício de *container*s e não trata execuções de longa duração. HAND (*Highly Avaiable Dynamic Deployment Infrastructure*) [Qi 2007] é um trabalho interessante que foi adotado como solução na versão mais recente do GT, a versão 4.2. HAND trata a publicação dinâmica de serviços no nível de *containers* e serviços. Apesar de prover melhorias no GT ainda obriga o reinício do *container* para que os serviços fiquem disponíveis. Esse é um inconveniente importante em execuções de longa duração, pois leva a uma fila de serviços esperando pela publicação enquanto as execuções do *container* estão sendo atendidas.

3. Uma Infra-estrutura para Execução Dinâmica de Serviços

A computação sob demanda requerida pelas Organizações Virtuais exige maior escalabilidade, flexibilidade e disponibilidade dos serviços. Para atender a esses requisitos é importante que a infra-estrutura da grade ofereça reconfiguração com a possibilidade de publicação de novos recursos ou atualização dos já disponíveis sem interromper as tarefas em execução. No GT4 os serviços são publicados manualmente de forma independente, mas são reconhecidos somente quando os *containers* são iniciados. *Containers* ativos devem ser reiniciados para que os novos serviços sejam disponibilizados ou para que os serviços já existentes possam ser atualizados. A última versão do GT (4.2) oferece recursos para automatizar a publicação de serviços, mas mantém a obrigatoriedade de reiniciar os *containers* para disponibilizar novos serviços.

Neste artigo mostramos uma infra-estrutura que agrega ao GT facilidades para execução de *workflows* com instanciação dinâmica de serviços. Ela faz a publicação de novos serviços quando necessário para a execução do *workflow*, ativa um novo *container* para que esses serviços fiquem imediatamente à disposição e coordena o uso dos *containers* criados desativando-os após o uso. A publicação dinâmica foi implementada através de um serviço e não requer modificações no GT4. Essa opção permite que a nossa solução possa ser usada em outras grades como o GT 4.2, que oferecem funcionalidades para a publicação dinâmica. A infra-estrutura é formada por um conjunto de serviços que oferece as seguintes funcionalidades:

Instanciação dinâmica de serviços: o usuário não é obrigado a identificar a localização do recurso computacional para a execução dos serviços do *workflow*. Durante a execução a infra-estrutura procura a melhor opção para executar cada um dos serviços;

Coordenação automática de referências (*endpoint reference service*): ao oferecer instanciação dinâmica, algumas atividades devem ser transparentes ao usuário. Por exemplo, um serviço executado gera um arquivo que é usado em serviços subseqüentes no *workflow*. Como a localização dos serviços é feita sob demanda, a infra-estrutura resolve as referências entre os serviços durante a execução não requerendo a participação do usuário;

Publicação dinâmica de serviços: ao identificar a melhor opção de recurso computacional a infra-estrutura pode publicar o serviço nesse recurso se for necessário. Isso é feito sem a necessidade de reiniciar o *container* padrão do GT; e

Execução mais robusta de *workflows*: se um serviço falhar durante a execução, a infra-estrutura pode pesquisar um recurso computacional alternativo, redirecionando automaticamente a execução com os devidos acertos nas referências. Se não houver um recurso com o serviço alvo disponível, a infra-estrutura tenta publicar o serviço em um dos recursos da grade. Somente após essas tentativas falharem é que a execução será interrompida.

4. A Arquitetura

A arquitetura da infra-estrutura proposta, formada por um conjunto de serviços, é mostrada na Figura 1. Os seus principais componentes são um gerente de *workflows*, um

escalonador, um serviço de publicação que realiza a publicação de serviços quando necessário e repositórios usados pelos componentes.

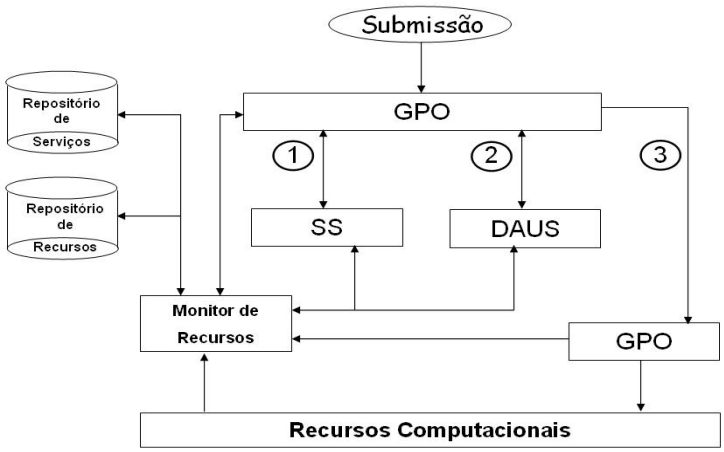

Figura 1. A arquitetura proposta.

Em uma execução típica, o usuário submete um *workflow* ao *Grid Process Orchestration* (GPO), o gerente de *workflows* do sistema. Este analisa o *workflow* consultando o serviço de escalonamento (*Scheduler Service* - SS) que identifica o melhor recurso disponível para cada serviço envolvido (1). Caso seja necessária a publicação do serviço, o GPO aciona o *Deployment And Undeployment Service* (DAUS) que é o responsável pela publicação do serviço. O DAUS publica o serviço retornando a sua localização (*endpoint reference*) ao GPO (2). De posse da localização do serviço, o GPO cria as instâncias necessárias durante a execução do *workflow* (3). Encerrada a execução o GPO automaticamente elimina as instâncias por ele criadas e aciona o DAUS para eliminar os serviços publicados e os *containers* iniciados. A seguir são detalhados os principais componentes da arquitetura.

4.1. O Gerente de *Workflows*

Em uma organização virtual, serviço passa a ser não só a base das aplicações como também a peça fundamental do processo de colaboração entre os participantes da organização. As composições de serviços trazem novas exigências notadamente no que se refere à coordenação e composição dinâmica de processos e serviços.

Para fazer a gerência das composições de serviços em nossa infra-estrutura foi usado o GPO [Senna 2009], um *middleware* para interoperabilidade de aplicações distribuídas que requerem composição de serviços em uma grade computacional. O GPO permite a criação e gerência de fluxos de aplicações, tarefas e, principalmente, serviços das grades computacionais. A arquitetura GPO é baseada nos conceitos apresentados em OGSA (implementação GT4), e estende a GJD (*WS-GRAM Job Description Language*) [Globus 2008] com as características WS-BPEL (*Web Service Business Process Execution Language*) [OASIS WS-BPEL 2006], uma especificação da OASIS (*Organization for the Advancement of Structured Information Standards*) na descrição dos fluxos a serem executados.

A Arquitetura GPO

A arquitetura GPO consiste de três componentes principais: GPO *Run*, GPO *Maestro Factory Service* e arquivos especificados com GPO *Language* (GPOL), apresentados na Figura 2.

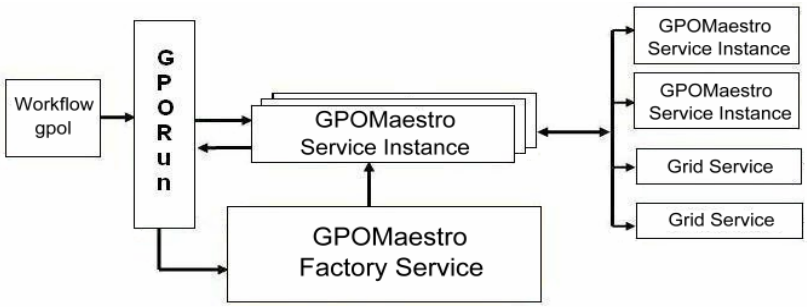

Figura 2. A arqtuitetura GPO.

GPO *Run* é o utilitário com o qual o usuário submete *workflows* ao GPO. GPO Maestro foi construído usando o conceito de fábrica/instância oferecido pelas grades computacionais. GPO *Run* usa o serviço GPO *Maestro Factory* para criar instâncias que ficam responsáveis pela execução do *workflow* que lhe é passado como argumento.

A *Grid Process Orchestration Language* (GPOL)

Na construção dos *workflows* usados no experimento foi usada a linguagem GPOL [Senna 2007]. A GPOL é uma linguagem baseada em XML que une facilidades da GJD e os conceitos de orquestração de serviços propostos em WS-BPEL. Foram acrescidas algumas diretivas específicas necessárias ao ambiente das grades, tais como: manutenção de estado, serviços potencialmente transientes, notificação, serviços orientados a dados e serviços orientados a grupos. GPOL inclui variáveis, ciclo de vida, controle fábrica/instância, controle de fluxo e tratamento de falhas. Adicionalmente, ela permite ao usuário iniciar execuções de tarefas, serviços e *workflows* de forma seqüencial ou em paralelo.

4.2. O Serviço de Apoio à Publicação Dinâmica de Serviços (DAUS)

Na execução do *workflow* o GPO consulta o escalonador sobre a melhor opção para executar o serviço. O escalonador pode responder indicando um recurso computacional onde o serviço não está disponível. Nesse caso o GPO aciona o DAUS que de posse da identificação do recurso computacional, executa as seguintes ações para publicar dinamicamente o serviço requerido pelo *workflow*:

Cria uma instância do DAUS no recurso remoto;

Consulta a instância remota sobre a necessidade de enviar ou não o código do serviço para o recurso remoto. Essa verificação garante que o código será transmitido somente uma vez durante a execução do *workflow*. Se o código ainda não estiver no recurso remoto, o DAUS obtém o código junto ao repositório de serviços e transmite-o;

- Requisita um número de porta ainda não usada pelos *containers* no recurso remoto;

- Consulta a instância remota sobre a necessidade de publicação do serviço, pois pode ocorrer do serviço estar publicado no recurso, mas não ser possível a criação de uma nova instância para atender a execução do *workflow*. Se necessário, a instância remota faz a publicação do serviço;

- Para não interromper os processamentos em execução que estão usando o *container* padrão e permitir o uso imediato do serviço no recurso remoto, a infra-estrutura inicia um novo *container* na porta disponível; e

- Com todas as ações executadas e o novo *container* disponível o DAUS entrega ao GPO a localização precisa do serviço (URI) no recurso remoto: Protocolo://IPHost:Porta/Nome_do_Serviço

O GPO usa a localização do serviço publicado para criar todas as instâncias do serviço necessárias para executar o *workflow*. Ao término da execução o GPO elimina as instâncias por ele criadas e aciona novamente o DAUS para eliminar o *container* criado e retirar os serviços publicados no recurso remoto.

4.3. O Serviço de Escalonamento (SS)

O serviço de escalonamento (SS) mostrado na Figura 1 tem a função de distribuir as tarefas do *workflow* a serem executadas nos recursos da grade. Para isso, o escalonador pode usar desde algoritmos simples como distribuir as tarefas conforme o número de núcleos dos processadores em cada recurso, até algoritmos mais complexos [Bittencourt 2008] que usam informações dos monitores da grade na tentativa de otimizar uma função objetivo.

As grades orientadas a serviço com instanciação dinâmica possuem peculiaridades que introduzem novas informações a serem utilizadas pelo escalonador, tais como:

- Tempo para criar a instância remota do DAUS responsável pela execução de tarefas no recurso computacional remoto;

- Tempo para obtenção de porta remota disponível para a criação do *container*;

- Tempo para transportar o código do serviço até o recurso remoto;

- Tempo para publicar o serviço no recurso remoto; e

- Tempo para iniciar um novo *container* que disponibiliza o serviço para uso imediato.

Essas novas informações podem ser incorporadas à função objetivo otimizada pelo escalonador visando, por exemplo, minimizar o tempo total de execução do *workflow* (*makespan*).

Neste trabalho focamos na avaliação dessas novas informações. Dessa forma, os resultados aqui apresentados poderão ser utilizados no desenvolvimento de escalonadores avançados em grades com instanciação dinâmica de serviços.

4.4. O Monitor de Recursos e os Repositórios

O monitor de recursos recebe informações sobre os recursos da grade e atualiza os repositórios do sistema. O repositório de recursos contém as características de cada recurso computacional e informações de desempenho e ocupação. Essas informações podem ser usadas pelo escalonador na tomada de decisão. O repositório de serviços contém as informações sobre os serviços disponíveis na grade e armazena os arquivos necessários para a publicação dinâmica dos serviços.

5. A Implementação e o Cenário de Aplicação

Com o objetivo de avaliar a infra-estrutura proposta construímos um protótipo formado por um conjunto de serviços desenvolvidos para o GT4. A infra-estrutura foi concebida e construída como um conjunto de três grupos de serviços, o gerente de *workflows* (GPO), o escalonador (SS) e o serviço de publicação dinâmica (DAUS). Os três grupos são independentes facilitando atualizações individuais e permitindo a substituição por soluções alternativas. Na implementação usamos o *Java* (*Sun*) acrescido do conjunto de APIs que formam o núcleo do GT4, disponível em todas as suas instalações. A seguir discutimos avaliações do protótipo implementado.

5.1. *Workflow* Executado em uma Grade Oportunista

Neste primeiro cenário executamos um *workflow* que usa três serviços em paralelo cujo resultado é usado em um quarto serviço. Esse *workflow* é executado em uma grade oportunista que permite o uso de recursos ociosos, mas é muito dinâmica com grande volume de entrada e saída de recursos.

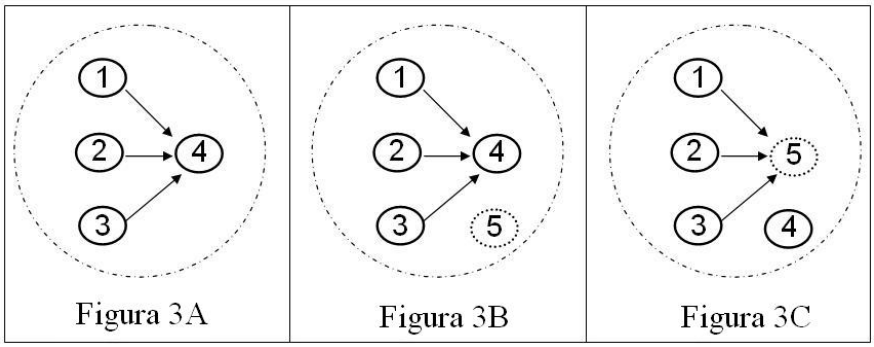

Figura 3. Workflow em uma grade oportunista.

Conforme mostra a Figura 3A, o *workflow* dispara a execução em paralelo nos recursos 1, 2 e 3, entregando os resultados ao recurso 4. O *workflow* concreto, com os recursos computacionais definidos, para esse grafo é mostrado na Figura 4. Ele está escrito em GPOL.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpo:job name="term-dyn">
  <gpo:definitions name="job_Definitions">
    <gpo:variables>
      <gpo:variable name="R_D1" type="double"/>
      <gpo:variable name="R_D2" type="double"/>
      <gpo:variable name="R_D3" type="double"/>
      <gpo:variable name="R_D4" type="double"/>
    </gpo:variables>
    <gpo:gservices>
```

```
      <gpo:gsh name="R_1"
            uri="https://10.3.77.15:8443/wsrf/services/ServiceA"
            type="Factory"
            return_type="double"/>
      <gpo:gsh name="R_2"
            uri="https://10.3.77.16:8443/wsrf/services/ServiceB"
            type="Factory"
            return_type="double"/>
      <gpo:gsh name="R_3"
            uri="https://10.3.77.19:8443/wsrf/services/ServiceC"
            type="Factory"
            return_type="double"/>
      <gpo:gsh name="R_4"
            uri="https://10.3.77.5:8443/wsrf/services/ServiceD"
            type="Factory"
            return_type="double"/>
    </gpo:gservices>
  </gpo:definitions>
<gpo:process name="job_Process">
  <gpo:flow>
      <gpo:invoke name="R_1" method="operacao1">
            <gpo:argument value="1" type="double"/>
            <gpo:return variable="R_D1" type="double"/>
      </gpo:invoke>
      <gpo:invoke name="R_2" method="operacao2">
            <gpo:argument value="2" type="double"/>
            <gpo:return variable="R_D2" type="double"/>
      </gpo:invoke>
      <gpo:invoke name="R_3" method="operacao3">
            <gpo:argument value="3" type="double"/>
            <gpo:return variable="R_D3" type="double"/>
      </gpo:invoke>
   </gpo:flow>
      <gpo:invoke name="R_4" method="operacao4">
            <gpo:argument variable="R_D1" type="double"/>
            <gpo:argument variable="R_D2" type="double"/>
            <gpo:argument variable="R_D3" type="double"/>
            <gpo:return variable="R_D4" type="double"/>
      </gpo:invoke>
  <!-- Return value to GPO_Client -->
  <gpo:return variable="R_D4"/>
</gpo:process>
</gpo:job>
```

Figura 4. Workflow GPOL concreto.

Em <*gpo:definitions*> são declaradas quatro variáveis de *workflows* (R_D1, R_D2, R_D3 e R_D4) que servem para manter o estado entre as invocações das operações do serviços. São definidos também quatro serviços ("/wsrf/services/ServiceA", ..., "/wsrf/services/ServiceD") a serem usados na execução. Cada serviço recebe um nome único que será usado na execução do *workflow* (R_1, R_2, R_3 e R_4). Note que os serviços estão precisamente definidos. Por exemplo, o serviço "/wsrf/services/ServiceA" está em https://10.3.77.15:8443 (protocolo:IP:porta).

Na Figura 3B vemos a entrada do recurso 5 na grade. Mesmo este recurso sendo melhor alternativa para a execução que o recurso 4, o *workflow* concreto não permite a otimização. Com a infra-estrutura que permite invocação dinâmica de serviços, podemos aproveitar o recurso 5. Com uma pequena simplificação na definição dos serviços no *workflow*, deixamos a escolha para a infra-estrutura.

```
      <gpo:gsh name="R_4" uri="/wsrf/services/ServiceD"
            type="Factory" return_type="double"/>
```

Figura 5. Definições flexíveis dos serviços.

A Figura 5 mostra que retiramos a localização do recurso computacional para o serviço "/wsrf/services/ServiceD", pois a linguagem GPOL aceita as duas formas para identificar o serviço. Ao identificar o serviço sem a localização do recurso computacional, o gerente de *workflow* GPO consulta o serviço de escalonamento SS sobre qual a melhor opção. Estando na situação da Figura 3B, com a entrada do recurso 5, o SS avalia os recursos 4 e 5 e informa a melhor opção ao GPO. Se o serviço "/wsrf/services/ServiceD" estiver pronto para uso no recurso escolhido, ou seja, publicado e o *container* ativo, o GPO cria a instância para a execução e redireciona automaticamente os resultados de R_1, R_2 e R_3 para a instância criada, continuando a execução do *workflow* (Figura 3C). Se o serviço não estiver disponível no recurso, o GPO aciona o DAUS para providenciar a publicação do serviço no recurso escolhido pelo SS e iniciar um novo *container* para atender ao processamento do *workflow*. O DAUS retorna a localização do serviço publicado (protocolo://IP:porta) e o GPO faz os redirecionamentos continuando a execução.

Para decidir qual a melhor opção de recurso computacional, o SS deve considerar os tempos envolvidos com a publicação dinâmica do serviço. Os tempos de publicação devem ser acrescidos ao desempenho dos recursos computacionais. Devido ao custo de publicação acrescido um recurso com melhor desempenho pode ser preterido caso haja um recurso menos potente, mas que já tenha o serviço pronto para o uso. Para avaliar os custos envolvidos com a publicação dinâmica executamos um experimento simples em uma grade.

5.2. Experimento para Avaliar os Custos da Publicação Dinâmica

As operações envolvidas com a publicação dinâmica de um serviço, descritas na Subseção 4.2 são: criação de instância local e remota do DAUS responsáveis pela operação de publicação, tempo para transmissão do código do serviço, tempo para obtenção da porta disponível no recurso remoto, tempo para executar o procedimento de publicação do serviço e o tempo para iniciar o novo *container* criado para atender ao novo serviço. Para avaliar esses tempos usamos uma grade com quatro recursos computacionais cujas características estão mostradas na Tabela 1. Nas execuções o GPO fica em uma das máquinas executando o *workflow*. Optamos pela escolha seqüencial do recurso computacional como estratégia de escalonamento (*round robin*).

Nome	Processador	Clock	Memória
Apolo	Intel Pentium IV 3.2 HT	3,2 Ghz	2,5 Gb
Zeus	Intel Pentium IV 3.2 HT	3,2 Ghz	4 Gb
Dionísio	Intel Core 2 Quad	2,4 Ghz	4 Gb
Cronos	Intel Core 2 Quad	2,4 Ghz	4 Gb

Tabela 1. Características dos recursos computacionais.

Sobre essa grade executamos algumas variações do *workflow* mostrado na Figura 4 usando serviços que fazem cálculos matemáticos simples. As variações são: em dyn0 todas as instâncias são locais, em dyn1 usamos três instâncias locais e uma é publicada dinamicamente, em dyn2 usamos duas instâncias locais e duas publicadas, em dyn3 usamos uma local e três publicadas e em dyn4 usamos quatro instâncias

publicadas dinamicamente. Nas Tabelas a seguir os valores correspondem sempre à média de pelo menos 1000 execuções de cada *workflow*. A Tabela 2 mostra os tempos médios para a execução desses *workflows*. Na coluna A estão os tempos onde todas as instâncias dinâmicas criadas fazem a cópia do código e a publicação do serviço. Na coluna B estão os resultados de uma opção de otimização do sistema, a cópia do código é feita somente na primeira publicação do serviço durante a execução do *workflow*.

	A	B
dyn0	2.733,15	2.743,90
dyn1	14.833,05	10.996,55
dyn2	27.130,05	13.294,20
dyn3	38.798,90	27.454,35
dyn4	51.270,60	35.306,55

Tabela 2. Tempos médios para a execução dos workflows (ms).

Os tempos detalhados por atividade executada na publicação estão na Tabela 3. Eles foram medidos sem nenhuma otimização e todas as atividades são sempre executadas, incluindo a cópia do código.

	Criação	Porta	iLocal	Código	iRemota	GDG	GSC
dionisio	11.352,50	150,08	15,78	276,70	32,66	2.626,08	7.522,58
cronos	11.510,64	139,58	14,84	277,64	28,20	2.595,00	7.416,88
apolo	9.927,88	124,72	18,36	274,02	20,70	2.606,84	7.427,58
zeus	9.666,62	141,84	14,74	275,72	20,90	2.578,74	7.519,42

Tabela 3. Tempos das atividades para a publicação dinâmica (ms).

Os tempos estão apresentados por recurso mostrando a variação de desempenho entre eles. A coluna Criação é o tempo total a partir do pedido do GPO ao DAUS até a recepção da referência pronta para o uso. Portanto, os tempos mostrados nessa coluna agregam os tempos mostrados nas outras colunas. A coluna Porta mostra o tempo gasto obtendo o número da porta remota. O DAUS usa controle distribuído das portas usadas, sendo cada recurso responsável pelo uso e liberação das portas usadas no recurso computacional onde está instalado. As colunas iLocal e iRemota mostram os tempos para a criação das instâncias do DAUS. A coluna Código mostra o tempo gasto transportando o código do serviço até o recurso. Em nosso experimento usamos serviços pequenos com códigos entre 40 e 100 Kb. A coluna GDG é o tempo gasto para a execução do *script* Globus que publica o serviço no recurso e a Coluna GSC é o tempo gasto para iniciar o *container* na porta escolhida, o que é feito através de um *script* Globus.

É possível notar que os tempos gastos com a infra-estrutura são mínimos. A maior parte do tempo é consumida pelos *scripts* Globus, sendo independente da infra-estrutura proposta e inerente a outras infra-estruturas que utilizam essa mesma tecnologia. Outro fator importante está relacionado ao código que usa tempo proporcional ao seu tamanho, cujo transporte pode ser feito uma única vez no *workflow* com a opção de otimização ativa.

De qualquer forma o tempo gasto para a criação da instância dinâmica, em torno de 11 s pela Tabela 3, é aceitável principalmente se comparado aos benefícios. O

workflow concreto requer que o usuário observe a grade para escolher os recursos disponíveis, sendo muito trabalhoso aproveitar as máquinas que entram e saem da grade oportunista. Além disso, se um dos recursos fixados no *workflow* concreto não estiver disponível não será possível a execução, enquanto que no *workflow* flexível, com instanciação dinâmica, o *workflow* será executado, com um acréscimo de tempo, mas será executado.

5.3. Outros Cenários

Um segundo cenário de aplicação mostra a infra-estrutura no tratamento de falha de um recurso computacional durante a execução do *workflow.* Ao identificar a falha o GPO solicita ao SS a indicação de um recurso disponível que possa substituir o recurso que falhou. Como primeira opção o SS tenta encontrar um recurso pronto com o serviço disponível. Se essa opção não estiver disponível ou exista uma opção melhor considerando os custos da publicação dinâmica, o SS indica um recurso que requer a publicação dinâmica do serviço necessário. Nessa situação o GPO usa o DAUS, que faz a publicação, iniciando um novo *container* para atender a execução. Recurso e serviços aptos o GPO continua a execução do *workflow*.

Ao terminar a execução o GPO sinaliza ao DAUS que pode eliminar o *container* e retirar o serviço. Esse esquema funcional eliminando o que foi criado durante a execução do *workflow* não altera as configurações dos recursos não sobrecarregando-os, pois o que foi criado ou instalado é eliminado, permanecendo o recurso como estava antes do uso.

Um terceiro cenário de aplicação mostra que prover serviços sob demanda pode melhorar o desempenho da grade mesmo com os tempos gastos com a publicação. Se em um *workflow* um serviço gera um arquivo que será usado na seqüência por outro serviço e esses serviços estão instalados em recursos computacionais distintos será necessário o transporte do arquivo entre os recursos computacionais. Como a infra-estrutura oferece a possibilidade de instalar serviços sob demanda, o escalonador pode decidir qual o menor custo para a execução mais eficiente. Se o tempo de publicação for menor que o tempo para transmissão do arquivo é conveniente publicar o serviço no recurso onde está o arquivo gerado.

Nesses cenários também é necessário redirecionar as entradas e saídas dos serviços, pois há uma substituição de recurso computacional que já foi comentada no cenário de grade oportunista anteriormente apresentado.

5.4. Analise Qualitativa

Qualquer aplicação de *workflow* pode se beneficiar da instanciação dinâmica de serviços com a coordenação automática de referências e publicação dinâmica de serviços. LIGO [LIGO 2008] e Montage [Montage 2008] são exemplos de *workflows* de aplicações reais que podem ser executadas com a infra-estrutura proposta. As melhorias acrescentadas ao GT4 pela nossa infra-estrutura são:

Escalabilidade: a infra-estrutura contribui ao permitir que novos recursos estejam disponíveis para uso, pois ao perceber a disponibilidade de novos recursos faz a instalação dos serviços preparando-o para o uso. O GT4 não oferece essa funcionalidade.

Flexibilidade: *workflows* com execução dinâmica de serviços adaptam-se perfeitamente à dinâmica da grade, pois os recursos são identificados no momento da execução de acordo com os recursos disponíveis;

Interoperabilidade: como a localização dos recursos é feita pela infra-estrutura, as execuções ficam restritas somente às políticas de uso dos vários domínios na organização virtual. Eventuais mudanças nas políticas não tem impacto nos *workflows*, pois estes adaptam-se às novas regras sem necessidade de alterações;

Complexidade: em *workflows* concretos o usuário fixa os recursos e fica com a obrigação de avaliar constantemente os recursos da grade para fazer as melhores escolhas. Usando *workflows* com execução dinâmica de serviços a infra-estrutura é responsável pelas decisões e pelo acompanhando da dinâmica da grade; e

Reuso: composições feitas com *workflows* flexíveis são mais versáteis e facilitam o reuso dos serviços.

6. Conclusão e Trabalhos Futuros

As grades computacionais baseadas em OGSA e WSRF oferecem suporte adequado para a heterogeneidade de recursos e a diversidade de aplicações. No entanto tem limitações para suportar a dinâmica desses ambientes.

Neste artigo apresentamos uma infra-estrutura que oferece provisionamento sob demanda na execução de *workflows* em grades computacionais, e mostramos uma implementação dessa infra-estrutura, que agrega a facilidade de execução dinâmica de serviços em *workflows* ao GT4, uma vez que este não oferece esse recurso. A adição dessas facilidades torna o ambiente das grades mais robusto, mais flexível aumentando a disponibilidade de recursos computacionais.

Discutimos os benefícios da infra-estrutura através de três cenários de aplicação comumente encontrados nas grades. Nas grades oportunistas a infra-estrutura melhora o aproveitamento preparando os recursos computacionais aumentando a sua disponibilidade. Um segundo cenário mostra que a infra-estrutura torna o ambiente mais robusto uma vez que permite novas alternativas para a correção de falhas durante a execução de *workflows*. E o provisionamento sob demanda com a publicação dinâmica de serviços abre novas possibilidades aos escalonadores melhorando o desempenho da grade, como foi discutido através do terceiro cenário de aplicação.

Como trabalhos futuros vamos analisar o uso de experimentos complexos com *workflows* dinâmicos para grades conectadas por redes ópticas.

7. Agradecimentos

Os autores gostariam de agradecer à CAPES, à FAPESP (05/59706-3) e ao CNPq (472810/2006-5 e 142574/2007-4) pelo apoio financeiro.

Referências

Bittencourt, L. F., Madeira, E. R. M. (2008) "A performance oriented adaptive scheduler for dependent tasks on grids". Concurrency and Computation: Practice & Experience, John Wiley & Sons, 20(9):1029-1049.

Byun, E., Kim, J. (2007) "DynaGrid: A dynamic service deployment and resource migration framework for WSRF-compliant applications", Parallel Computing, Vol. 33 Issues 4-5, February 20, pp. 328-338.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S. (2003) "The next step in web services," Communications of ACM, vol. 46, no. 10, pp. 29–34.

Foster, I., Kesselman, C., and Tuecke, S. (2002) "The physiology of the grid: An open grid services architecture for distributed systems integration", [Online] http://www.globus.org/research/papers/ogsa.pdf.

Globus Alliance (2008) "Globus Toolkit Version 4 – GT4", http://www.globus.org/toolkit/.

Goldchleger, A., Kon, F., Goldman, A., Finger, M. (2003) "InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines", ACM/IFIP/USENIX International Workshop on Middleware for Grid Computing. Rio de Janeiro, Brazil, June.

LIGO (2008) Laser Interferometer Gravitational Wave Observatory, www.ligo.caltech.edu/.

Montage (2008) An Astronomical Image Mosaic Engine, http://montage.ipac.caltech.edu/.

OASIS WS-BPEL (2006) "Web Services Business Process Execution Language Version 2.0", Public Review Draft, http://docs-oasis-open.org/wsbpel/2.0/, Agosto.

OASIS WSRF (2006) "Web Services Resource Framework v1.2", http://www.oasis-open.org/specs/index.php#wsrfv1.2.

Qi, L., Jin, H., Foster, I., Gawor, J. (2007) "HAND: Highly Avaiable Dynamic Deployment Infrastruture for Globus Toolkit 4", in Proc. 15th Euromicro Int. Conf. Parallel, Distributed and Network-Based Processing (PDP 07), IEEE CS Press, pp. 155-162.

Senna, C. R. (2007) "GPO – Um Middleware para Orquestração de Serviços em Grades Computacionais", Dissertação de Mestrado, UNICAMP – Universidade de Campinas, São Paulo, Brasil, Fevereiro.

Senna, C. R., Bittencourt, L. F., Madeira, E. R. M. (2009) "Execution of Service Workflows in Grid Environments" , in Proc. 5th Int. Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Washington, USA, April 6-8, 2009.

Sun, H., Zhu, Y., Hu, C. et al. (2005) "Early Experience of Remote and Hot Service Deployment with Trustworthiness in CROWN Grid", Advanced Parallel Processing Technologoies, LNCS, Springer Berlin / Heidelberg, pp. 301-312.

UFCG (2008) OurGrid Home Page. www.ourgrid.org.

Watson, P., Fowler, C. (2005) "An Architecture for the Dynamic Deployment of Web Services on a Grid or the Internet", Technical Report Series, CS-TR-890, School of Computing Science, Newcastle University, February, 2005.

Weissman, J., Kim, S., England, D. (2005) "A Framework for Dynamic Service Adaptation in the Grid: Next Generation Software Program Progress Report", in Proc. 19th IEEE Int. Parallel and Distributed Processing Symp. (IPDPS 05), IEEE CS Press, pp 5-9.