On-line Collection of Monitoring Data in Distributed Systems Using a Hierarchical Structure

Rafael Keller Tesser, Lucas Mello Schnorr, Philippe O. A. Navaux

Instituto de Informática,
Universidade Federal Do Rio Grande do Sul
{rktesser, lmschnorr, navaux}@inf.ufrgs.br

Abstract

Several tasks performed by computer systems can profit from the use of monitoring information. In distributed systems, one needs to gather the data that is scattered throughout the system components. Only then, it's possible to correlate information from different resources. In this paper we present a hierarchical model for the collection of monitoring data in distributed systems. Its goal is to enable on-line analysis of the behavior of system and programs, using visualization. The model is composed of three kinds of components: collectors, aggregators and clients. The collectors get the monitoring data that is generated locally, in the resources, and send them to a set of aggregators. These forward this data to higher level aggregators or clients, forming a hierarchical structure. The clients provide the data they receive to a component that will do processing and analysis tasks. The collection model also includes a subscription mechanism that allows the client to select the collectors from which it wants to receive data. This reduces the unnecessary transmission of information. A prototype implementation was made, in order to evaluate the model in a real distributed system. A modified (to allow on-line updating) DIMVisual prototype has been used to accomplish the integration of the monitoring data. The visualization is provided by TRIVA, with a module that reads data from the DIMVisual prototype. This paper presents results of an evaluation of data transmission times using the prototype.

1. Introdução

Sistemas distribuídos podem tirar grande proveito do uso de ferramentas de monitoramento. Elas podem fornecer várias informações sobre o sistema, como o estado de seus componentes, dados sobre a execução de aplicações ou sobre o funcionamento da rede de interconexão. Os dados de monitoramento podem ser usados para múltiplas tarefas, como detecção de falhas, escalonamento, depuração, análise de desempenho e adaptação de aplicações. É possível também armazenar dados históricos a fim de analisar o comportamento do sistema ao longo de um determinado período de tempo, por exemplo: durante a execução de um programa.

Grid é um tipo especial de sistema distribuído que possui características intrínsecas que apresentam desafios para o monitoramento. Dentre elas destacam-se: grande escala, distribuição geográfica, dinamismo da disponibilidade de recursos e a heterogeneidade dos mesmos. Portanto, é uma plataforma interessante para pesquisa na área de monitoramento.

O presente trabalho propõe um modelo hierárquico para coleta de dados de monitoramento distribuídos. Seu objetivo é possibilitar a realização de análise comportamental de sistemas e programas distribuídos, de maneira *on-line*, através de visualização. *On-line*, no contexto deste trabalho, significa que os dados são atualizados progressivamente, em tempo de execução, em contraste com a análise *post-mortem*. Esse modelo foi projetado para trabalhar em conjunto com o modelo de integração de dados *DIMVisual* [16]. Devido a essa ligação, o modelo de coleta foi nomeado *DIMVisual Hierarchical Collection Model* (DIMVHCM).

O DIMVHCM é composto por três tipos de componentes: coletores, agregadores e clientes. Os coletores, locais aos recursos, obtém os dados de diversas ferramentas de monitoramento. Agregadores recebem dados dos coletores e de outros agregadores e os repassam aos interessados. Os agregadores são organizados hierarquicamente, sendo que os dados recebidos são enviados para agregadores de nível mais alto até chegarem aos clientes. Esses últimos repassam os dados para objetos que realizam o processamento e análise dos dados. O modelo utiliza um mecanismo de inscrições que possibilita que os dados sejam enviados somente aos componentes que os requisitarem.

Um protótipo do DIMVHCM foi implementado, para sua avaliação e validação. Nesse protótipo os dados coletados são fornecidos ao protótipo do DIMVisual. Esse último foi modificado para suportar a atualização *on-line* dos dados. Além disso, foi utilizado a ferramenta TRIVA [14, 15], para gerar visualizações das informações. Para isso, foi implementado um módulo de leitura para o TRIVA poder receber, de maneira *on-line*, dados integrados pelo protótipo do *DIMVisual*.

Foram realizados experimentos com o protótipo, a fim de verificar a duração dos envios de dados entre coletores e um cliente. Para isso foram utilizadas diferentes organizações hierárquicas e configurações dos coletores.

A seção a seguir apresenta uma pequena introdução ao monitoramento de sistemas distribuídos. A seção 3 trata do monitoramento de *Grid* e apresenta alguns trabalhos relacionados ao assunto. Em especial, a subseção 3.1 fala sobre algumas ferramentas de monitoramento utilizadas em *Grid*. A seção 4 aborda detalhes do *DIMVisual*. Na seção seguinte, tem-se uma introdução sobre o TRIVA. O DIMVHCM é apresentado com mais detalhes na seção 6. A seção 7 aborda aspectos da implementação do protótipo. Os experimentos citados acima são assunto da seção 8. Por fim, conclui-se o artigo e apresentam-se tarefas a serem realizadas futuramente no âmbito do trabalho.

2. Monitoramento de Sistemas Distribuídos

O uso de ferramentas de monitoramento em sistemas distribuídos permite a obtenção de dados relacionados ao estado do sistema e de seus componentes, e também sobre a execução de aplicações. Essas ferramentas fornecem tanto informações estáticas, como a configuração do sistema, quanto dinâmicas, como uso de CPU ou memória disponível. Além disso, podem ser fornecidos dados sobre a execução de programas, através da instrumentação dos mesmos, de maneira a gerar rastros de sua execução.

Os dados de monitoramento podem ser utilizados para auxiliar em várias tarefas [3, 9]. Entre elas estão a descoberta de recursos, escalonamento, depuração, seleção de réplicas, adaptação de aplicações, análise de desempenho e detecção de falhas.

Uma dificuldade encontrada em sistemas de monitoramento distribuídos é a coleta dos dados de monitoramento de maneira que possam ser fornecidos às aplicações interessadas em seu uso. Para isso deve-se ter uma arquitetura que possibilite a localização desses dados e sua transmissão.

3. Monitoramento de Grid

Nessa seção são apresentados alguns trabalhos relacionados ao monitoramento de *Grid*. É dada atenção especial a ferramentas de monitoramento, as quais são assunto da subseção 3.1. Para cada uma são apresentadas diferenças em relação ao DIMVHCM.

Um grupo de trabalho do Global Grid Forum (GGF) produziu uma especificação de alto nível para uma arquitetura de monitoramento de *Grid*, chamada *Grid monitoring architecture* (GMA) [18]. A figura 1 mostra os componentes da GMA: produtores de informação, consumidores e serviço de diretório. Os produtores de informação registram-se no serviço de diretório e os consumidores executam consultas ao mesmo para descobrir quais informações estão disponíveis e obter os dados necessários para acessar os produtores. Portanto, na GMA os dados trafegam diretamente dos produtores para os consumidores.

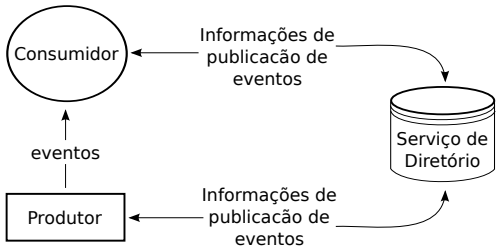

Figura 1. Componentes da GMA

Zanikolas e Sakellariou criaram uma taxonomia de sistemas de monitoramento de *Grid* [20]. Ela classifica-os em três níveis, de acordo com a presença de publicadores e republicadores e suas características. O DIMVHCM possui publicadores (coletores) e republicadores (agregadores) que podem ser distribuídos arbitratiamente em uma hierarquia. Isso enqudra-o no nível 3 dessa taxonomia.

3.1 Ferramentas de Monitoramento

Ferramentas de monitoramento de *Grid* necessitam boa escalabilidade, devido ao grande número de recursos a serem monitorados. Além disso, devem levar em conta o dinamismo da disponibilidade dos recursos e a heterogeneidade dos mesmos. Nos próximos parágrafos são abordadas algumas ferramentas de monitoramento utilizadas nesse tipo de sistema.

Ganglia [11, 13] é um sistema de monitoramento projetado inicialmente para uso em *clusters* de computadores. Apesar disso, é muito comum o sua utilização em *Grid*. Sua arquitetura, representada na figura 2, funciona através de dois tipos de serviços. O primeiro constitui-se de serviços de coleta de dados de monitoramento que executam localmente a cada recurso, chamados *gmond*. O segundo tipo, chamado *gmetad*, provê acesso aos dados para os interessados em utilizá-los. Os *gmetad* podem ser organizados hierarquicamente na forma de árvore, formando uma federação de sistemas, possibilitando assim acesso a informações de

diversos sistemas federados. Apesar de utilizar uma estrutura hierárquica, os *gmetad* utilizam *polling* para a detectar a disponibilidade de novos dados. No DIMVHCM utiliza-se um mecanismo baseado em eventos em que o componente que fornece os dados é quem inicia seu envio aos componentes inscritos como interessados nesses. Outra diferença é que o Ganglia não possui um mecanismo de inscrição para recebimento de dados.

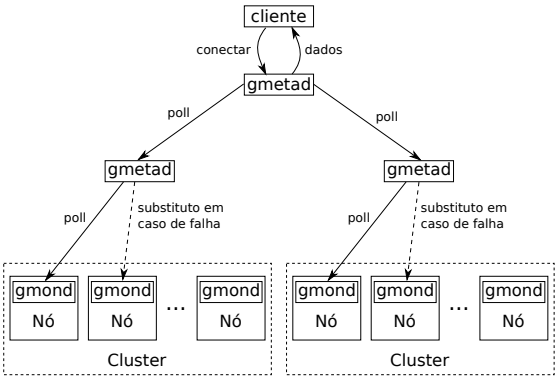

Figura 2. Arquitetura do Ganglia

O *Monitoring and Discovery Service* (MDS) [3] é um serviço de monitoramento e descoberta de recursos que faz parte do *Globus Toolkit* [17]. O MDS versão 2 utiliza uma estrutura hierárquica para descoberta e transmissão dos dados. O MDS versão 4 [6, 5] apresenta uma estrutura simplificada em relação ao MDS2, utiliza *web services* e incorpora suporte a inscrição para recebimento de notificações. No entanto, esse serviço é voltado principalmente para a descoberta de recursos, tomada de decisão baseada no estado do sistema e detecção e notificação de falhas baseada em eventos.

A ferramenta GridICE [2, 1] possui uma arquitetura estruturada em cinco camadas. A primeira é o Serviço de Medição, responsável por coletar medidas dos recursos. Os dados coletados são armazenados em um repositório local do *site*. O segundo é o Serviço de Publicação, responsável por oferecer os dados coletados a seus consumidores. Esse serviço faz uso do GIS (*Grid Information Service*) que é um componente do Globus MDS2. A terceira camada é o Serviço de Coleta de Dados que permite a coleta de dados históricos. A quarta camada corresponde aos serviços de detecção e notificação e ao serviço de análise de dados. O primeiro é responsável por notificar a ocorrência de eventos. O segundo fornece análise de desempenho, do nível de utilização, relatórios e estatísticas em geral. A última camada corresponde ao serviço de apresentação, que é uma interface gráfica *web*. Uma diferença em relação ao DIMVHCM é que, devido a basear-se no MDS2, essa ferramenta não suporta a inscrição para recebimento de dados. Além disso, apenas informações de registro são propagadas pela hierarquia do GIS, os dados são obtidos diretamente da fonte através de uma URL fornecida no seu registro.

MonALISA [12, 10] é um sistema desenvolvido para prover um *framework* de informações de gerenciamento aplicado a tarefas de processamento de dados da física de altas energias. Essa ferramenta utiliza a tecnologia JINI da linguagem Java e *web services*. A MonALISA permite acesso a diversas ferramentas de monitoramento existentes. A descoberta de recursos é feita através de *JINI Lookup Services* (LUS). Há também clientes, que permitem a visualização das informações coletadas. O foco está ligado ao gerenciamento do sistema, apresentando uma interface *on-line* que permite a visualização de dados sobre os recursos e a rede de interconexão. É interessante notar que, apesar de poder receber dados de várias ferramentas diferentes, eles são apresentados de forma não normalizada. Assim como o DIMVHCM, essa ferramenta suporta inscrição para o recebimento de dados, mas as aplicações clientes conectam-se diretamente aos serviços dos quais querem receber dados.

NetLogger [7] é um conjunto de ferramentas para monitoramento do comportamento de aplicações distribuídas. Sua utilização dá-se através da modificação de componentes da aplicação e também do sistema operacional para que registrem eventos de interesse. Esses eventos são correlacionados com o comportamento do sistema com o objetivo de caracterizar seu desempenho e da rede.

O NetLogger Activation Service [8] permite o controle do nível de instrumentação do NetLogger. Seu objetivo é melhorar a escalabilidade, para uso em *Grid*. Esse mecanismo também possibilita a inscrição para recebimento de dados de monitoramento, no entanto não é especificada uma maneira de formar uma estrutura hierárquica de coleta.

O *Network Weather Service* (NWS) [19] é uma ferramenta que tem por objetivo prover informações históricas e previsão do desempenho da rede e dos computadores em um sistema distribuído. Essa ferramenta foi desenvolvida para utilização por escalonadores e fornecimento de dados de *quality-of-service*. Em comparação ao DIMVHCM, além da diferença de objetivos, o NWS possui uma arquitetura centralizada com relação a um serviço de diretório e não suporta inscrição para recebimento do dados.

4. DIMVisual

O *DIMVisual* [16] é um modelo de integração de dados de monitoramento para visualização. Nesse modelo, os dados são coletados por diferentes ferramentas de monitoramento. Depois, eles passam por um processo de integração e são unificados para poderem ser visualizados em conjunto, através de uma ferramenta de visualização.

O processo de integração, representado na figura 3, tem três etapas: sincronização, unificação e padronização. A

sincronização é necessária devido aos dados provirem de diferentes origens. A segunda etapa consiste em unificar os identificadores utilizados pelas ferramentas de monitoramento conforme uma hierarquia de entidades (cluster, nó, processo, etc.). O último passo, consiste em converter a forma de registro de dados de cada ferramenta de coleta para um formato padrão que possibilite sua visualização.

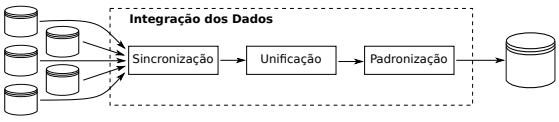

Figura 3. Modelo da integração de dados no *DIMVisual*

Existe um protótipo do *DIMVisual* que recebe arquivos de dados coletados por várias ferramentas e gera um arquivo visualizável utilizando a ferramenta Pajé [4]. Esse protótipo tem por componentes principais as fontes de dados, o ordenador de dados, o controlador de integração e o controlador de aplicação.

As fontes de dados possuem um leitor, um sincronizador e um conversor. O primeiro é responsável pela leitura dos arquivos. O segundo corrige os tempos dos dados. O terceiro é responsável pela unificação dos identificadores e pela padronização dos dados. O ordenador gerencia as fontes de dados. Ele lê o evento com menor tempo dentre os disponíveis de todas as fontes. O controlador de integração faz a unificação hierárquica dos dados, fornece aos conversores os identificadores das entidades utilizadas, configura o sistema e grava em arquivo os resultados fornecidos pelo ordenador. Por fim, o controlador de aplicação fornece ao usuário uma interface para a configuração do protótipo.

5. TRIVA

TRIVA (*Three dimensional Interactive Visualization Analysis*) um protótipo de ferramenta de visualização que implementa um conjunto de abordagens para visualização de aplicações paralelas. Essas abordagens incluem a utilização de visualização em três dimensões e de treemaps.

Na abordagem em três dimensões o eixo vertical é utilizado para representar, através de barras, a evolução das entidades observadas no tempo. Interações entre entidades podem ser representadas através de linhas ligando-as. As outras duas dimensões são utilizadas para formar uma base de visualização em que são agregadas informações sobre recursos. Essa abordagem permite observar interações entre processos, assim como a relação da aplicação com a plataforma de execução.

A abordagem treemap agrupa as entidades de acordo com informações de localização (por sites, por clusters, por nós, etc.), de maneira hierárquica. As medidas são representadas por retângulos, cujo tamanho relativo representa seu valor. É possível modificar o nível hierárquico da visualização. Em um determinado nível, os valores representados são uma sumarização dos valores referentes ao nível logo abaixo. Por exemplo: a medida de uso de CPU para um cluster pode ser a soma ou a média desse valor nos nós que o compõem.

Além disso, o TRIVA possui suporte à agregação dos dados por fatia de tempo (*time-slice*). Essa técnica consiste da sumarização dos valores medidos durante um intervalo de tempo. Esse intervalo pode ser movido, permitindo observar o comportamento da aplicação.

6. Modelo de Distribuição de Coletores de Dados de Monitoramento

Em sistemas distribuídos há a necessidade de reunir os dados de monitoramento que são gerados em seus componentes. Só assim é possível integrá-los de maneira a poder-se correlacioná-los. Uma abordagem para isso é reunir todos os dados em armazenamento local. Nesse caso, após o fim do processo observado (*post-mortem*) esses dados são reunidos para a leitura das informações. Outra abordagem é a coleta progressiva dos dados de monitoramento durante a execução. Dessa maneira, pode-se analisar as informações antes do fim do processo. Isso é útil, por exemplo, quando se está observando um processo que executa por um tempo longo. Nesse caso, anomalias podem se corrigidas assim que observadas. Não há necessidade de esperar o fim da execução.

Nesse trabalho propõe-se o DIMVHCM, que é um modelo para coleta de dados de monitoramento distribuídos. Os dados são coletados de forma *on-line* e fornecidos para uma ferramenta de monitoramento. Esse modelo foi projetado para integrar-se ao *DIMVisual*, que seria responsável por integrar e unificar os dados provenientes de diversas origens e ferramentas. Além disso, o DIMVHCM foi desenvolvido tendo como foco a visualização *on-line* do comportamento de sistemas e programas distribuídos.

O DIMVHCM possui três tipos de componentes: coletores, agregadores e clientes. Os coletores ficam situados nos recursos a serem monitorados e são responsáveis por ler os dados obtidos pelas ferramentas de monitoramento e enviá-los aos agregadores. Esses últimos são organizados hierarquicamente e recebem dados de monitoramento de um conjunto coletores ou agregadores de nível inferior. Essas informações são repassadas para agregadores de nível superior ou para clientes. Um cliente é responsável por fornecer os dados a componentes que irão realizar as etapas seguintes do monitoramento.

A estruturação hierárquica dos agregadores é adequada para o uso em *Grid* devido ao mesmo ser geralmente or-

ganizado dessa maneira. Por exemplo, um Grid pode ser formado por um conjunto de *sites*, que por sua vez possui um conjunto de clusters formados por um conjunto de nós.

Esse modelo é genérico quanto ao formato dos dados de monitoramento, assim como o *DIMVisual*. O suporte a um determinado formato é uma questão de implementação. Para cada formato é necessário implementar um coletor capaz de obter os dados de seu gerador e um componente que receba-os de um cliente e seja capaz de utilizá-los.

O modelo utiliza-se de um mecanismo de inscrição que possibilita que os dados fornecidos por um coletor só sejam transmitidos aos componentes interessados. O funcionamento desse mecanismo é descrito na subseção 6.5.

6.1. Coletores

Um coletor é um componente responsável por coletar os dados obtidos por uma ferramenta de monitoramento. Por esse motivo, deve haver um coletor específico para cada ferramenta. Os dados coletados devem ser enviados periodicamente aos agregadores inscritos para recebê-los. Os coletores possuem um conjunto definido de agregadores nos quais devem registrar-se para poder enviar dados para os mesmos. Esse registro permite e que os agregadores possam identificá-los e inscrever-se como interessados em seus dados. A subseção 6.4 aborda o registro dos coletores.

6.2. Agregadores

Os agregadores formam uma estrutura hierárquica de transmissão dos dados de monitoramento. Eles recebem dados de coletores ou de outros agregadores e os repassam para os agregadores de nível superior interessados nos mesmos. Isso é feito de maneira que um cliente possa adquirir dados sobre todo o sistema a partir dos agregadores situados no topo da hierarquia.

Para cada conjunto de recursos é comum ter-se uma máquina, ou um pequeno grupo dessas, servindo como ponto de ligação com os recursos de outros agrupamentos. Levando em conta essa característica, poderia-se ter os agregadores localizados nesses nós, fornecendo acesso aos dados de monitoramento do conjunto de recursos.

Vale ressaltar que o modelo não restringe o número de agregadores que podem receber dados de uma mesma origem. Dessa maneira pode-se implementar um mecanismo em que se tenha múltiplos agregadores expondo diferentes conjuntos de dados. Então, através de mecanismos externos de controle de acesso aos agregadores, poderia-se ter diferentes níveis de acesso aos dados de monitoramento. Uma aplicação para isso seria permitir que um *site* forneça informações mais detalhadas para clientes internos, e menos para os externos ao seu conjunto de recursos.

6.3. Cliente

O cliente é responsável por fornecer os dados de monitoramento recebidos a um componente que irá processá-los e analisá-los. O receptor dos dados têm de conhecer o formato dos mesmos. Portanto, é necessário no mínimo um desses para cada formato de dado recebido. O cliente seleciona o componente a quem deve encaminhar os dados através de um tipo associado a cada coletor. Um exemplo de de tal receptor seria uma fonte de dados do *DIMVisual*.

Na sua inicialização o cliente deve solicitar a um agregador os registros dos coletores que podem ser acessados através desse. Dessa maneira o cliente passará a conhecer as fontes de dados disponíveis (através do tipo do coletor), os recursos dos quais pode-se obter dados e os identificadores dos coletores. Esses identificadores são utilizados nas solicitações de inscrição e para saber a origem dos dados de monitoramento recebidos.

6.4. Registro dos Coletores

Na sua inicialização os coletores possuem um conjunto de agregadores aos quais devem registrar-se. Ao receber uma solicitação de registro de um coletor, um agregador cria uma entrada para o mesmo em um registro de coletores. Cada um desses registros contém o identificador do coletor, sua localização (recurso monitorado), seu tipo e a sua origem (coletor ou agregador de onde os dados serão recebidos). O identificador servirá para identificar os dados enviados pelo coletor e auxiliará nas buscas no registro de coletores. A localização e o tipo de coletor são úteis para os clientes identificarem os recursos e as fontes de dados. Além disso, como já citado, o tipo é usado pelo cliente para saber a quem repassar os dados de monitoramento enviados por um coletor. A origem ajuda no atendimento a solicitações de inscrição em coletores segundo um processo explicado mais adiante, possibilitando saber através de quem deve-se repassar a solicitação. Caso haja agregadores de nível hierárquico superior conectados ao agregador, então os dados de registro dos coletores devem ser repassados a eles, apenas com a origem modificada para ser o agregador que está repassando os dados.

Esse registro possibilita que agregadores não precisem conhecer previamente os componentes de nível hierárquico inferior. Dessa maneira pode-se adicionar e remover coletores sem reiniciar os agregadores. Além disso, pode-se adicionar agregadores à estrutura da mesma maneira, com a restrição de que componentes devem ser iniciados anteriormente aos que pretendem enviar dados para eles.

6.5. Inscrição Para Recebimento de Dados

Para diminuir a transmissão desnecessária de dados que não interessam à aplicação cliente, tanto coletores quanto

agregadores devem enviar dados apenas a agregadores ou clientes registrados como interessados nos mesmos. Para isso o modelo proposto possui um sistema de inscrição que permite selecionar os coletores dos quais pretende-se receber informações.

Um cliente ou um outro agregador pode fazer uma inscrição para recebimento de dados de monitoramento. Ao recebê-la, o agregador registra o requerente como interessado nos dados. Caso o agregador ainda não esteja inscrito para recebimento dos dados solicitados, ele solicita a inscrição para um de nível inferior que possa acessar os dados, ou caso já esteja no nível mais baixo, para um coletor. Isso se repete nos agregadores de nível inferior até que se atinja um agregador que já esteja inscrito para receber os dados ou então o coletor que os fornece. Cada agregador ao receber uma requisição de inscrição registra o requerente em um conjunto de interessados nos dados do coletor, armazenado no agregador. Assim também, os coletores registram o agregador que fez a requisição como interessado nos dados. Para solicitar uma inscrição um agregador fornece, além de sua identificação, o identificador do coletor.

Quando não houver mais interesse nos dados por um agregador ou cliente, esse deve cancelar sua inscrição. Ao receber a solicitação de cancelamento, o agregador retira o solicitante do conjunto de interessados no coletor. Caso não haja outros interessados nos dados referentes à requisição ele então solicita o cancelamento de sua inscrição por não haver mais a necessidade de receber tais dados. Assim, as solicitações de cancelamento propagam-se em direção à base da hierarquia. Caso não se encontre no caminho um agregador com outras inscrições relacionadas ao coletor a que a solicitação refere-se, a mesma chegará ao coletor. Esse deve, então, remover inscrição do agregador adequado.

A figura 4 mostra um exemplo de hierarquia em que se tem dois *clusters*, cada um com um agregador que pode receber dados dos coletores situados em seus nós. As circunferências C1, C2 e C3, representam 3 tipos de coletores. Os agregadores armazenam registros dos coletores a que têm acesso, contendo um identificador e um conjunto de destinos para os quais devem encaminhar os dados recebidos daquele coletor. Isso é representado na figura pelas listas ligadas por linhas tracejadas aos agregadores. Da mesma maneira, cada coletor possui uma lista dos agregadores para os quais deve enviar seus dados. No exemplo da figura, o agregador 1, identificado por A1, está inscrito no coletor C2 do nó 1 do *cluster* 1, identificado por C2-CL1-N1. Ao receber dados do referido coletor ele verifica seu registro do mesmo. Fazendo isso ele descobre que o agregador 3 está inscrito para receber os dados e os envia ao mesmo. O agregador 3, por sua vez possui uma inscrição de um cliente do DIMVHCM, ao qual ele encaminha os dados.

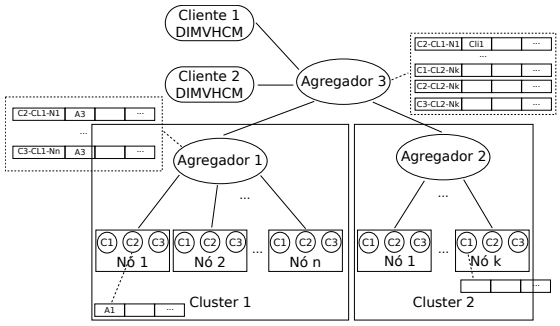

Figura 4. Exemplo de hierarquia do modelo proposto

7. Protótipo do DIMVHCM

A figura 5, apresenta os componentes utilizados na implementação de um protótipo do DIMVHCM. Primeiro, tem-se o DIMVHCM em si, que fornece dados no formato da fonte em que os obteve para o protótipo do *DIMVisual*. Esse, por sua vez, integra os dados dados e os devolve convertidos ao formato da ferramenta de visualização Pajé. Então, tem-se o *DIMVHCM Reader*, que é um módulo do TRIVA, responsável por sua ligação com o *DIMVisual*. No restante da seção abordamos aspectos da implementação dos componentes do DIMVHCM e também as adaptações realizadas no *DIMVisual* e no TRIVA.

Inicialmente, foram implementadas classes base, que permitem a construção de coletores agregadores e clientes personalizados. Essa implementação foi feita utilizando objetos distribuídos da biblioteca base da linguagem Objective-C. Tanto *DIMVisual* quanto TRIVA são implementados nessa linguagem. As classes implementadas realizam as tarefas que não dependem da fontes dos dados. Entre essas estão gerenciamento de conexões, do registro de coletores e de subscrições. Essas classes são chamadas `DIMVCollector`, `DIMVAggregator` e `DIMVClient`.

No protótipo, um objeto `DIMVClient` fornece os da-

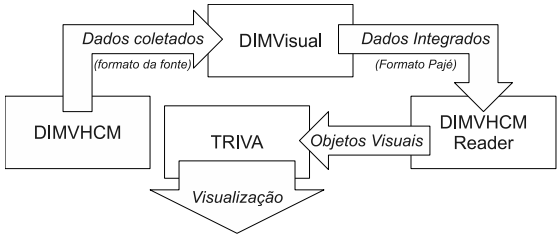

Figura 5. Componentes do protótipo do DIMVHCM

dos recebidos para uma fonte de dados do *DIMVisual*. Esse último, então, integra e padroniza os dados, conforme são recebidos. No entanto, o protótipo preexistente, funcionava apenas de maneira *post-mortem*, lendo os dados de arquivos. Portanto, foi necessário modificá-lo para suportar o fornecimento e integração *on-line* dos dados.

Quanto ao TRIVA, foi preciso implementar um módulo leitor que utiliza o *DIMVisual* como fonte de dados. Esse módulo utiliza o modelo produtor-consumidor. Uma thread recebe os dados do *DIMVisual* e outra os fornece ao módulo responsável pela etapa seguinte do processamento do TRIVA. Além disso, o consumidor tem que ativar a atualização do intervalo de visualização quando da chegada de novos dados. Além disso, foi adicionada uma interface gráfica que permite o gerenciamento de inscrições aos coletores. Para esse protótipo utilizou-se a abordagem de visualização com treemaps.

8. Avaliação dos componentes

Nessa seção são apresentadas medições do tempo de envio de dados de monitoramento dos coletores até um cliente. Para obtenção dos resultados foram utilizados oitenta computadores de um cluster. Os nós estão equipados, cada um, com dois processadores dual core de 2.6 GHZ, 4 GB de memória e estão conectados por Gigabit Ethernet. Em cada nó foi executado um coletor. Esse obtém os dados no formato XML de um *gmond* do Ganglia. Esses testes avaliam apenas os componentes do DIMVHCM. Não foi utilizado o componente de visualização.

Foram medidos os tempos de envio de dados utilizando três diferentes organizações da hierarquia do DIMVHCM. A primeira consiste de todos os coletores enviando os dados coletados para um mesmo agregador, que os envia para um cliente. Na segunda, os computadores formam quatro grupos de vinte membros. Cada grupo envia dados para um diferente agregador que os repassa ao cliente. A última organização consiste da segunda adicionada de um agregador intermediário entre os que recebem os dados dos coletores e o cliente, adicionando um nível na hierarquia. Nos resultados apresentados a seguir, essas organizações são nomeadas *1ag. + 1cli*, *4ag + 1cli* e *4ag + 1ag + 1cli*, respectivamente.

Além disso, as medições foram feitas com quatro configurações dos coletores, quanto à frequência das coletas e ao agrupamento dos envios. As configurações são denotadas nos resultados por P*t*NL*n*, onde *t* é o período em segundos entre cada coleta e *n* é o número de coletas agrupadas por envio. Por exemplo, P1NL5 significa que é realizada uma coleta a cada 1 segundo e que os dados são enviados a cada 5 coletas (aproximadamente a cada 5 segundos). Para cada uma das doze combinações de configurações, coletou-se os tempos durante aproximadamente 30 minutos

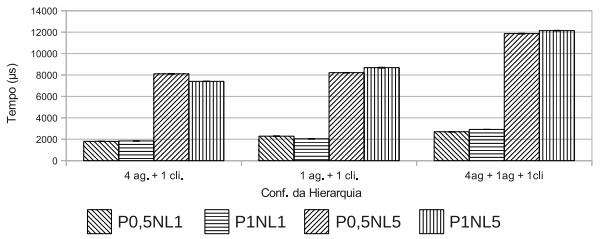

Figura 6. Duração média dos envios entre coletores e cliente

de execução.

A figura 6 mostra a média dos tempos por envio. Pode-se notar que, para 80 coletores, não houve grande diferença nos entre as hierarquias *4 ag + 1 cli* e *1 ag + 1 cli*. A organização *4ag + 1ag + 1cli* apresenta maiores tempos devido nível hierárquico extra. Ressalta-se que todas as médias são bastante inferiores ao intervalo entre os envios.

Para melhor comparação entre as configurações dos coletores, o gráfico da figura 7 apresenta os valores do anterior divididos pelo numero de coletas agregadas em cada envio. Por ele pode-se notar que a agregação dos dados apresenta uma pequena diminuição nos tempos de envio. Também percebe-se que, para a quantidade de coletores usada, a duplicação da frequência dos envios fez pouca diferença.

9. Conclusão e trabalhos futuros

Sistemas distribuídos podem utilizar dados sobre o estado e funcionamento de seus componentes e aplicações para a realização de várias tarefas. Daí vem a utilidade de ferramentas de monitoramento para obtenção dessas informações. Uma tarefa envolvida nesse processo é a coleta de dados de monitoramento que estão espalhadas pelo sistema.

Esse trabalho propôs um modelo, chamado DIMVHCM, para coleta de dados de monitoramento distribuídos. Esse modelo tem por objetivo fornecer dados para a análise

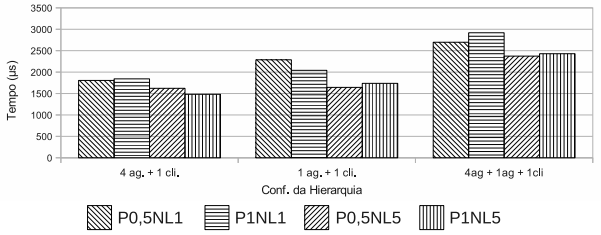

Figura 7. Duração média dos envios entre coletores e cliente por coleta

on-line do comportamento de sistemas e aplicações distribuídos, através de visualização. A arquitetura desse modelo é composta por coletores, agregadores e clientes. Os coletores são responsáveis por obter as informações dos recursos e enviá-las para os agregadores. Os agregadores são organizados hierarquicamente e repassam os dados recebidos para agregadores de nível superior ou para clientes. Clientes repassam os dados recebidos para componentes que realizam seu processamento e análise. Além disso, o modelo inclui um mecanismo de inscrição, que possibilita a transmissão dos dados referentes a um coletor apenas aos componentes registrados para recebê-los.

Foi implementado um protótipo do DIMVHCM, incorporando o *DIMVisual*, para integração de dados e o TRIVA, como ferramenta de visualização. Utilizando esse protótipo realizaram-se testes para avaliar tempos de envio de dados, com diferentes configurações da hierarquia e dos coletores.

Prováveis futuros trabalhos incluem testes com um maior número de coletores e variando o tamanho dos dados coletados. Além disso, deve-se fazer avaliações utilizando o componente de visualização. Também seria interessante avaliar a intrusividade do monitoramento.

Referências

[1] C. Aiftimiei, S. Andreozzi, G. Cuscela, G. Donvito, V. Dudhalkar, S. Fantinel, E. Fattibene, G. Maggi, G. Misurelli, and A. Pierro. Recent evolutions of gridice: a monitoring tool for grid systems. In *GMW '07: Proceedings of the 2007 workshop on Grid monitoring*, pages 1–8, New York, NY, USA, 2007. ACM.

[2] S. Andreozzi, N. D. Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, and M. C. Vistoli. Gridice: a monitoring service for grid systems. *Future Gener. Comput. Syst.*, 21(4):559–571, 2005.

[3] K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. Foster. Grid information services for distributed resource sharing. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, page 181, Washington, DC, USA, 2001. IEEE Computer Society.

[4] J. C. de Kergommeaux and B. de Oliveira Stein. Pajé : an extensible environment for visualizing multi-threaded programs executions. *Euro-Par 2000 Parallel Processing, Proc. 6th In-ternational Euro-Par Conference*, 1900:133–140, 2000.

[5] I. Foster. A globus toolkit primer, 2005.

[6] I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. In H. Jin, D. A. Reed, and W. Jiang, editors, *NPC*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2005.

[7] D. Gunter, B. Tierney, B. Crowley, M. Holding, and J. Lee. Netlogger: A toolkit for distributed system performance analysis. In *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 267, Washington, DC, USA, 2000. IEEE Computer Society.

[8] D. Gunter, B. L. Tierney, C. E. Tull, and V. Virmani. On-demand grid application tuning and debugging with the netlogger activation service. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 76, Washington, DC, USA, 2003. IEEE Computer Society.

[9] J. Hollingsworth and B. Tierney. Instrumentation and monitoring. In I. Foster and C. Kesselman, editors, *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[10] I. C. Legrand, H. B. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobreand. Monalisa: An agent based, dynamic service system to monitor, control and optimize grid based applications. In *Computing in High Energy and Nuclear Physics conference (CHEP2004)*, page 181, Interlaken, Switzerland, 2004. IEEE Computer Society.

[11] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30, 2004.

[12] H. B. Newman, I. C. Legrand, , P. Galvez, R. Voicu, and C. Cirstoiu. Monalisa: A distributed monitoring service architecture. In *Computing in High Energy and Nuclear Physics conference (CHEP2003)*, La Jola, California, 2003.

[13] F. D. Sacerdoti, M. J. Katz, M. L. Massie, and D. E. Culler. Wide area cluster monitoring with ganglia. In *2003 IEEE International Conference on Cluster Computing. Proceedings.*, pages 289–298, 2003.

[14] L. M. Schnorr, G. Huard, and P. O. A. Navaux. Towards visualization scalability through time intervals and hierarchical organization of monitoring data. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 428–435, Washington, DC, USA, 2009. IEEE Computer Society.

[15] L. M. Schnorr, G. Huard, and P. O. A. Navaux. Triva: Interactive 3d visualization for performance analysis of parallel applications. *Future Gener. Comput. Syst.*, 26(3):348–358, 2010.

[16] L. M. Schnorr, P. O. A. Navaux, and B. de Oliveira Stein. Dimvisual: Data integration model for visualization of parallel programs behavior. *ccgrid*, 00:473–480, 2006.

[17] The Globus Alliance. Página do globus toolkit, 2008. Disponível em: <http://www.globus.org/toolkit/>. Acesso em: Julho 2010.

[18] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolsky. A grid monitoring architecture, 2002. Disponível em: <http://www.ogf.org/documents/GFD.7.pdf>. Acesso em: Julho 2010.

[19] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

[20] S. Zanikolas and R. Sakellariou. A taxonomy of grid monitoring systems. *Future Generation Computer Systems*, 21(1):163–188, January 2005.