DASE – Distributed data Agent Service Environment

Fábio Silva Carvalho
Escola Politécnica da
Universidade de São Paulo
fabio.s.carvalho@poli.usp.br

Hélio Crestana Guardia
Universidade Federal de São Carlos
Departamento de Computação
helio@dc.ufscar.br

Fabio Aparecido Gamarra Lubacheski
Escola Politécnica da
Universidade de São Paulo
fabio.lubacheski@poli.usp.br

Líria Matsumoto Sato
Escola Politécnica da
Universidade de São Paulo
liria.sato@poli.usp.br

Resumo

Sistemas multi-agentes devem oferecer recursos suficientes para que seus agentes possam interagir de maneira satisfatória e atingir seus objetivos. Um exemplo de recurso é um conjunto de dados armazenados em algum tipo de mecanismo de persistência, como um sistema gerenciador de banco de dados. O acesso a dados deve ser possível mesmo que eles estejam distribuídos, fato inclusive que também caracteriza os sistemas multi-agentes. Assim, este trabalho apresenta um sistema cujo objetivo é prover a agentes o acesso a dados distribuídos de forma simples e transparente, ou seja, independentemente da complexidade que o ambiente dos agentes possui e das peculiaridades do Sistema de Banco de Dados Distribuído.

1. Introdução

Agentes são entidades bem definidas que possuem sociabilidade e um variado nível de autonomia e pró-atividade. Além dos próprios agentes, outros elementos compõem os sistemas multi-agentes: o ambiente que os mantém, uma coleção de objetos que interagem com eles, e um conjunto de regras que regulamenta todas as suas atividades. Os agentes, apesar de possuírem uma percepção limitada do ambiente que os cerca, devem ter acesso aos recursos necessários para atingirem seus objetivos. Tais recursos são oferecidos por outros agentes ou pelo próprio ambiente [1].

Sistemas multi-agentes são utilizados em diversos domínios de aplicações diferentes, principalmente os relacionados à solução de problemas complexos e de natureza distribuída, envolvendo dados, controle ou conhecimento distribuídos [2]. Alguns exemplos disso são sistemas de controle de tráfego aéreo [3], simulações de sistemas financeiros, gerenciamento de comunicações de redes [4], comércio eletrônico [2], sistemas hospitalares [5] e simulações de sistemas naturais e sociais complexos [6].

Em todos os exemplos citados os agentes precisam resolver problemas, interagir com o ambiente e comunicar-se com outros agentes e/ou usuários, atividades que envolvem a necessidade de representar o conhecimento sobre o ambiente externo. Além disso, o conjunto de dados que compõem a base deste conhecimento é descentralizado [7].

Assim, o acesso a dados armazenados em um ambiente distribuído é um requisito importante para qualquer sistema multi-agente. Entretanto, é fundamental que tal recurso seja utilizado de maneira padronizada e eficiente, assim como todo serviço oferecido dentro de uma comunidade de agentes.

Este trabalho[1] visa apresentar um sistema cujo objetivo é prover a um sistema multi-agente acesso a um ou mais sistemas de banco de dados (SBD). O ponto forte deste sistema será sua integração com um sistema multi-agente mesmo que tal ambiente, assim como o SBD, seja distribuído, inclusive garantindo a consistência dos dados independentemente de o SBD oferecer tal recurso.

O artigo está organizado da seguinte maneira: na segunda seção a idéia principal do sistema é proposta.

[1] Trabalho financiado pelo Projeto GIGA RNP - projeto 2468 - FINEP/FUNTTEL

Na terceira seção a plataforma de agentes adotada é descrita sucintamente e sua escolha é justificada. A quarta seção apresenta as principais características do sistema. A arquitetura do DASE é descrita na quinta seção. A sexta seção traz alguns aspectos funcionais do sistema e contém um exemplo de uso. Em seguida, a sétima seção traz resultados obtidos através de testes. Na oitava seção um breve levantamento a cerca da existência ou não de trabalhos semelhantes é realizado. O artigo termina com a apresentação de conclusões, além de relatar itens importantes para o aperfeiçoamento futuro do DASE.

2. Proposta

O sistema descrito neste artigo chama-se DASE, *Distributed data Agent Service Environment*. Seus dois principais objetivos são:

1. Prover a agentes de um sistema multi-agentes acesso a dados distribuídos de forma simples e transparente.
2. Oferecer recursos adicionais relacionados ao acesso aos dados, como, por exemplo, a garantia de consistência de dados através de controle de concorrência distribuído.

Embora os dois itens acima não sejam suficientemente detalhados para representar toda a funcionalidade que o sistema se propõe a oferecer, já é possível notar que o DASE não pretende ser uma plataforma de execução de agentes.

A sua proposta é ser um ambiente que oferece um conjunto de serviços que se integra de forma harmônica aos demais agentes de uma plataforma de agentes já existente. Portanto, o DASE é parte de um sistema multi-agente. Os serviços de seus agentes são relacionados ao acesso a dados distribuídos de um modo transparente, bem definido, eficiente e em conformidade com qualquer padrão que a plataforma de agentes se comprometa a seguir.

A seção seguinte traz a plataforma de agentes adotada e a justificativa de sua escolha.

3. Plataforma de agentes adotada

Existem alguns sistemas, tanto comerciais quanto acadêmicos, que oferecem uma plataforma de execução de agentes, ou, em outras palavras, um *middleware* [8] de agentes. Alguns estão em fase inicial de desenvolvimento, e outros se restringem a somente um determinado tipo de sistema multi-agente, ou então são voltados apenas a simulações [9-11].

Entretanto, duas plataformas destacam-se por oferecerem um ambiente de execução de agentes funcional, que trazem consigo diversas ferramentas auxiliares, e já são utilizadas tanto pela indústria, quanto pelo meio acadêmico.

JACK [12] é um *framework* de agentes desenvolvido em Java e que implementa a arquitetura BDI, *Belief-Desire-Intention* [13]. O JACK não é restrito a nenhuma linguagem específica de comunicação entre agentes, dando suporte inclusive a KQML, *Knowedge Query Manipulation Language* [14], embora tenha sido projetado para priorizar a comunicação baseada em troca de mensagens e ORB, *Object Request Broker* [15]. Esta decisão de projeto reflete a busca de maior compatibilidade com a indústria, como por exemplo, a tentativa de aproveitar a extensa adoção do paradigma orientado a objetos, e a necessidade de interoperabilidade entre sistemas.

Outra plataforma de agentes relativamente madura e robusta é o JADE [16]. O JADE também é desenvolvido totalmente em Java, assim como o JACK, e é compatível com as especificações FIPA2000, *Foundation for Intelligent Physical Agents* [17]. Assim, possui uma implementação dos agentes AMS, *Agent Management System*, e DF, *Directory Facilitator*, que representam respectivamente os serviços de páginas brancas e amarelas, de acordo com o padrão FIPA. Além de ser um ambiente de execução, possui também ferramentas de controle, teste e monitoração de agentes. O JADE é um projeto *open-source* e é mantido por um consórcio formado por algumas grandes empresas de tecnologia de informação e telecomunicações, além de ter o suporte de sua comunidade de usuários.

A plataforma escolhida para dar suporte ao DASE foi o JADE. Os principais motivos desta escolha são apresentados a seguir:

1. É desenvolvida em Java, o que garante portabilidade, fácil integração com sistemas distribuídos, além de todas as vantagens que o paradigma OO oferece.
2. É um projeto *open-source*, o que favorece a formação de uma comunidade de usuários participativos e a evolução do projeto de um modo descentralizado e transparente, além de ser mais coerente com o contexto de um projeto de cunho acadêmico.
3. Está relativamente madura, estável e possui um conjunto de funcionalidades, ferramentas e documentação bastante razoável.
4. Possui a preocupação de estar em conformidade com as especificações FIPA, o que certamente

favorece a padronização e a evolução da tecnologia.

4. Principais características

Antes de listar as características do DASE, é necessário definir os seguintes termos a serem utilizados ao longo do texto:

- **Agentes clientes**: agentes do sistema multi-agente que se relacionarão diretamente ou indiretamente com o DASE.
- **Ambiente de dados**: conjunto de informações referentes à identificação de um SBD específico cujos dados serão disponíveis aos agentes clientes através dos serviços prestados pelo DASE.

Assim como o JADE, o DASE também é desenvolvido usando Java. A lista a seguir contém as principais características do DASE.

1. Os serviços são disponibilizados através de agentes em conformidade com o padrão FIPA.
2. Qualquer complexidade inerente ao acesso aos dados é abstraída, o que garante a transparência durante o acesso aos dados.
3. A localidade física dos dados em acesso é transparente aos agentes clientes mesmo se os dados estiverem distribuídos.
4. Acesso a dados pelos agentes clientes de maneira padronizada independentemente do tipo ou versão do gerenciador de recursos.
5. Compatibilidade com diversos gerenciadores de recursos diferentes, desde que possuam *driver* JDBC, *Java Database Connectivity* [18].
6. Suporte simultâneo a múltiplos SBDs diferentes.
7. Acesso aos dados pelos agentes clientes sempre através de requisição SQL ou alguma variação adotada pelo gerenciador de recursos.
8. Suporte a controle de concorrência distribuído.
9. Suporte a balanceamento de carga.
10. API, *Application Program Interface*, eficiente, completa e bem documentada.
11. Interface gráfica com o usuário que permite a configuração global do sistema.
12. Suporte a registro e carregamento de ambientes de dados através de arquivos XML.

5. Arquitetura

O DASE interage com três subsistemas distintos, os agentes clientes, os agentes JADE e o SBD. Os agentes clientes utilizam serviços tanto de agentes DASE quanto de agentes JADE. Os agentes DASE também utilizam serviços de agentes JADE, além de utilizar os serviços do SBD. A figura 1 lustra a interação entre estes subsistemas.

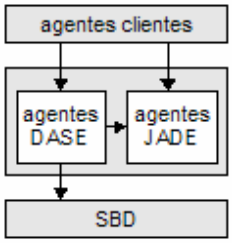

Figura 1. Interação entre subsistemas.

Os serviços disponíveis pelo JADE aos agentes clientes e aos agentes DASE são basicamente controle do ciclo de vida, identificação, publicação e localização de serviços, e infra-estrutura de comunicação entre os agentes. Os serviços providos pelos agentes DASE são localizados pelos agentes clientes graças à publicação no agente JADE DF.

5.1. Agentes DASE

Os agentes DASE são divididos em três grupos diferentes, apresentados a seguir.

5.1.1. Agentes do núcleo do sistema. São os principais agentes DASE, responsáveis por iniciar, controlar e monitorar o sistema.

(DM) *DASEManager*. É o principal agente DASE, sendo responsável pela configuração inicial do sistema e criação do agente DC. Permite o carregamento, início e encerramento de ambientes de dados, representados por agentes DE. Tem papel importante na API do DASE. Só há um agente DM em todo o sistema.

(DC) *DASEConfigurator*. Representa a única interface gráfica do DASE com o usuário, e oferece recursos de configuração global do sistema e controle de ambientes de dados, permitindo operações como criação, remoção, modificação e monitoração de ambientes de dados. O agente DC ainda traz informações extras sobre o sistema. Só há um agente deste tipo em todo o sistema.

5.1.2. Agentes de serviço de acesso a dados. São os agentes que lidam, diretamente ou indiretamente, com os agentes clientes, por isso compõem uma parte importante da interface de todo sistema. Suas tarefas estão relacionadas à representação de ambientes de dados e serviço de requisição de dados.

(DE) *DASEDataEnvironment*. Representa e controla um ambiente de dados. Sua principal função é

registrar informações sobre quais são os nós DASE participantes, e características sobre o SBD utilizado, balanceamento de carga e controle de concorrência. É responsável pelo controle dos agentes ND. Todo ambiente de dados deve ter pelo menos um nó, e, conseqüentemente, um agente ND. Como múltiplos ambientes de dados são disponibilizados simultaneamente, mais de um agente deste tipo pode existir.

(SL) *DASEServiceLocator*. É o agente requisitado pelo agente cliente para encontrar um agente DP. Este agente representa o *service locator* [19] presente em qualquer arquitetura orientada a serviços. É o principal agente responsável pelo balanceamento de carga do DASE. Existe um agente SL para cada agente DE.

(ND) *DASENode*. Representa um nó de um SBD e um nó do JADE para um determinado ambiente de dados. É responsável pelo controle dos agentes CP, DPM, DB e CC de seu nó. Registra informações sobre o endereço do nó, além de dados estatísticos e de controle, como controle de concorrência e balanceamento de carga, de acordo com especificações de seu ambiente de dados. Existe um agente ND para cada nó de um ambiente de dados. Um nó pode abrigar mais de um agente ND, já que mais de um ambiente de dados pode fazer uso daquele nó.

(DPM) *DataProviderManager*. É o responsável por gerenciar os agentes DP de um nó de um ambiente de dados. Determina quantos agentes DP devem existir, qual deverá atender uma requisição de dados, qual deve morrer se necessário, quando criar novos agentes DP, e quantos criar. Participa do balanceamento de carga do sistema. Existe um agente DPM para cada agente ND. Mantém sempre no mínimo um agente DP.

(DP) *DASEDataProvider*. Comunica-se diretamente com os agentes clientes recebendo suas requisições de serviço de acesso a dados. Tem papel importante durante o processo de controle de concorrência, atuando como gerenciador de transações. Este agente é baseado no padrão de projeto *Façade* [20].

5.1.3. Agentes de manipulação de dados. São os agentes que não são visíveis aos agentes clientes, mas que desempenham importantes tarefas para garantir que as requisições de dados sejam corretamente realizadas. Estes agentes estão mais próximos do SBD.

(CC) *ConcurrencyController*. É o principal agente responsável pelo controle de concorrência oferecido pelo DASE, atuando como escalonador de transações. Caso o controle de concorrência esteja ativado em um ambiente de dados, cada um de seus nós terá um agente CC, o que significa exatamente um agente CC para cada agente ND.

(CP) *ConnectionProvider*. Guarda qualquer tipo de informação necessária para conectar-se ao nó do SBD e é o único agente capaz de estabelecer uma conexão direta com o SBD. Presta serviços ao agente DB. Existe um agente CP para cada agente ND.

(DB) *DataBridge*. Além do agente CP, é o único que se relaciona diretamente com o SBD. Entretanto, somente este tipo de agente é capaz de enviar solicitações e receber dados diretamente do SBD. Sua função é repassar ao SBD toda instrução SQL que receber. Tem papel importante durante o processo de controle de concorrência, atuando como gerenciador de dados. Existe um agente DB para cada agente ND.

A figura 2 traz um diagrama que representa a relação entre os agentes descritos. A notação utilizada neste diagrama para o relacionamento entre agentes é semelhante à utilizada em um diagrama de classes UML.

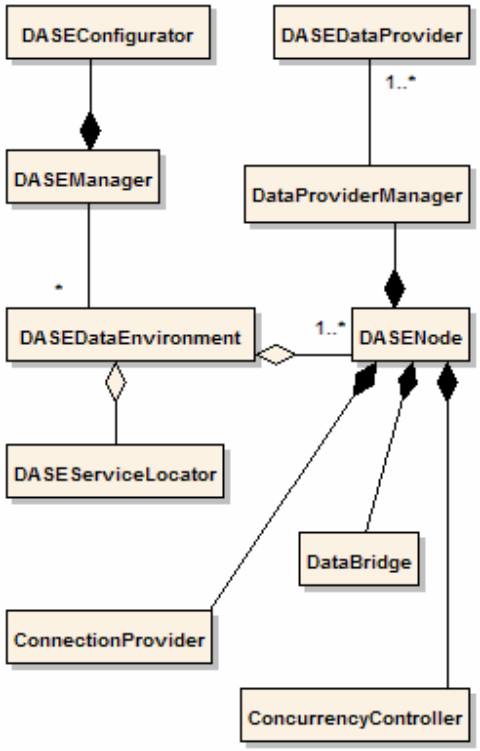

Figura 2. Relacionamento entre agentes DASE.

A figura 3 ilustra a arquitetura do sistema considerando um ambiente de dados disposto sobre dois nós, separados na figura através de uma linha tracejada vertical. Nesta figura os círculos representam agentes. Os agentes “ac” são os agentes clientes, enquanto que todos os demais são agentes DASE. Os

agentes JADE foram omitidos nesta figura apenas para simplificar a ilustração.

Os retângulos cinzas representam o conjunto de agentes DASE, um em cada nó. Os dois tambores, na parte inferior da figura, representam dois nós do SBD distribuído acessado pelos agentes clientes através do DASE. As setas pretas representam em seqüência a comunicação entre os agentes durante uma requisição de dados realizada por um agente cliente. Algumas setas são bidirecionais, portanto são numeradas duas vezes. A seta mais grossa e não numerada entre os agentes "cc" dos dois nós representa a interação entre eles durante o controle de concorrência distribuído.

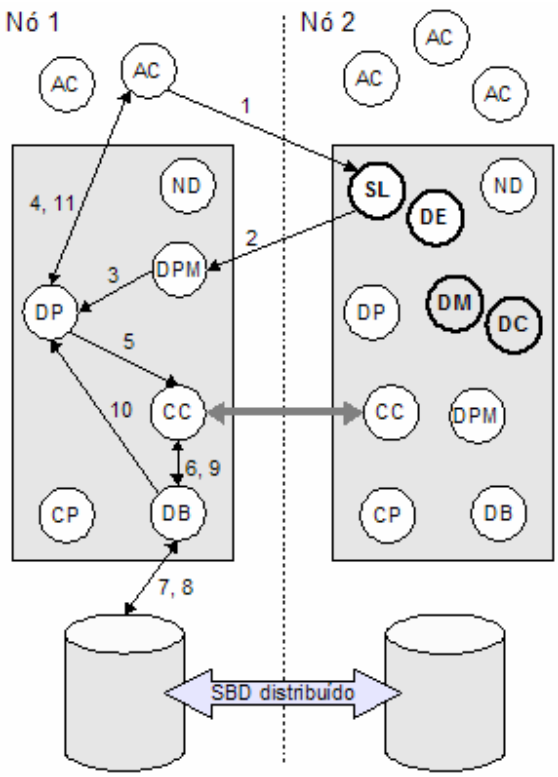

Figura 3. Arquitetura DASE com dois nós.

5.2. Controle de concorrência

O controle de concorrência garante a consistência dos dados mesmo que diversas transações os acessem. Quando considerado um SBD distribuído então, o controle de concorrência passa a possuir complexidade adicional. O DASE possui suporte transacional de dois modos diferentes:

1. Encaminhamento de transações ao SBD: o SBD dá suporte a transações e irá recebê-las, por isso o DASE simplesmente as encaminha.
2. Controle de concorrência distribuído do DASE: cada transação recebida pelo DASE será processada garantindo acesso consistente aos dados.

Os dois modos estão em conformidade com o padrão JTA, *Java Transaction API* [21].

O controle de concorrência distribuído do DASE é realizado através de diferentes técnicas que podem ser adicionadas ou removidas, como *plugins*. Dois exemplos dessas técnicas são *Two Phase Lock* e *Timestamps* [22]. O controle de concorrência do DASE é ativado e configurado através do ambiente de dados, e é realizado pelos agentes:

- *DASEDataPovider*: atuam como gerenciadores de transações.
- *ConcurrencyController*: atuam como escalonadores de transações.
- *DataBridge*: atuam como gerenciadores de dados.

O agente DP é o responsável por receber requisições de dados de agentes clientes, que podem estar em dois formatos: transações ou sentenças de acesso a dados independentes. Se o controle de concorrência estiver ativado, este agente repassará a requisição ao agente CC, caso contrário as repassará diretamente ao agente DB. O agente DP sempre receberá uma resposta diretamente do agente DB, contendo ou não resultados. Enquanto o aguarda, o agente DP permanece em estado bloqueado.

O agente CC recebe transações do usuário através de agentes DP e as transforma em transações do sistema. Pode receber também instruções de acesso a dados independentes, neste caso transformará cada uma delas em uma transação do sistema. Tal agente atua como escalonador de transações escolhendo a melhor ordem para execução de suas instruções. Em seguida, repassa as instruções ao agente DB.

Os agentes CC realizam controle de concorrência distribuído ao trabalhar em conjunto com agentes CC de outros nós do mesmo ambiente de dados. Eles suportam diferentes tipos de mecanismos de controle de concorrência a partir de diferentes tipos de comportamentos (*behaviors* JADE [23]) implementados e disponíveis.

Os agentes DB sempre retornam os resultados das instruções SQL ao agente DP, independentemente de o agente CC ter intermediado esta solicitação. Caso o controle de concorrência esteja ativado, sempre que o agente DB concluir o repasse de uma instrução SQL ao SBD, ele avisará o agente CC para que este fique ciente de que aquela instrução, que certamente compõe uma transação, já foi concluída e se obteve sucesso ou não.

Uma descrição detalhada sobre o controle de concorrência do DASE seria demasiado longa, devendo ser apresentada futuramente.

5.3. Balanceamento de carga

O DASE possui suporte a balanceamento de carga, equilibrando as requisições recebidas pelos nós de um ambiente de dados. Este recurso é ativado e configurado através do ambiente de dados.

O balanceamento de carga do DASE funciona da seguinte forma: o agente cliente localiza através do agente JADE DF o agente SL referente ao ambiente de dados desejado. O agente SL deve encaminhar ao agente cliente um agente DP. Cada nó do ambiente de dados possui agentes DP, e é através da escolha deste agente que o balanceamento de carga ocorre.

No entanto, o agente SL não escolhe diretamente o agente DP, e sim um nó. Para isso dois fatores são considerados. Primeiro é dada prioridade ao próprio nó de onde partiu a requisição. Segundo são consideradas consultas a cada agente DPM de cada nó do ambiente de dados. A partir dessas informações o agente SL decide se é viável escolher o próprio nó do agente cliente, o que ocorre na maioria dos casos, ou escolher um outro nó menos sobrecarregado. Depois que isto ocorre, o agente DPM alocará um agente DP disponível para atender à solicitação do agente cliente.

Cada agente DPM mantém uma informação de controle chamada *Load Number* (LN) que é utilizada durante o balanceamento de carga e é calculada considerando o número requisições pendentes em um nó. Duas observações são importantes quanto ao balanceamento de carga do DASE:

5.3.1. Indicação de nós. O agente SL aceita requisições de agentes clientes contendo a indicação de um ou mais nós. Assim, o nó escolhido será um dos indicados, condicionando o balanceamento de carga.

5.3.2. Técnica de distribuição dos dados. O balanceamento de carga do DASE considera que a localidade física dos dados distribuídos (replicados e segmentados) ao longo dos nós do SBD não é importante, o que ocorre quando a distribuição é feita utilizando distribuição circular (*round robin*), por *hashing*, ou qualquer outra técnica que não considere a semântica dos dados durante a distribuição, como por faixa de valores.

Esta restrição é importante porque não será vantajoso direcionar carga a um determinado nó se ele não tiver as melhores condições de acesso aos dados, ou mesmo se ele não contiver os dados. Pelo fato de o DASE não conhecer detalhes sobre a localidade dos dados do SBD, o balanceamento de carga é desativado por padrão. Assim, é assumido que as requisições devam ser processadas no nó em que foram submetidas, e a responsabilidade de escolher o melhor nó é passada aos agentes clientes.

O balanceamento de carga do DASE ainda está em fase de implementação, e será tratado de forma mais detalhada em futuras publicações.

6. Aspectos funcionais

Os resultados das requisições de acesso a dados são representados sempre através de objetos da classe *DASEResult*. Esta classe é abstrata e as subclasses concretas *IntegerResult*, *JDBCResult* e *StringResult* são utilizadas para guardar os dados. Cada uma representa um tipo de retorno específico, previsível ou não. *IntegerResult* é utilizado para requisições que não retornem dados ou que realizem modificações. *StringResult* é utilizado para requisições que retornem dados em formato de vetor de Strings. Já *JDBCResult* é utilizado para requisições que retornem um objeto JDBC *ResultSet*. A decisão entre *StringResult* e *JDBCResult* fica a cargo do agente cliente.

6.1. Exemplo de uso

De acordo com a plataforma JADE, um agente possui um ou mais comportamentos, que são semelhantes a tarefas. O código da figura 4 representa um comportamento de um agente cliente cuja tarefa é criar um novo ambiente de dados DASE. Ainda seria necessário criar nós para este ambiente de dados, para depois ativá-lo, o que é feito através de uma solicitação ao agente *DASEManager*.

```
public void action() {

  Runtime rt = Runtime.instance();
  Profile p = new ProfileImpl();
  ContainerController cc;
  cc = rt.createAgentContainer(p);

  Object arguments = {
    jdbcDriverClass, jdbcUrl, user,
    password, cc};

  AgentController de;

  try {

    de = cc.createNewAgent(
      "DASEDataEnvironment",
      "dase.DASEDataEnvironment",
      arguments);

    de.start();
```

```
  } catch (StaleProxyException e) {
    e.printStackTrace();
  }
  completed = true;
  myAgent.doDelete();
}
```

Figura 4. Criação de um ambiente de dados.

A requisição de dados de um agente cliente é feita a um agente DASE DP através de troca de mensagens. O JADE suporta diversos protocolos FIPA para interação entre agentes. O trecho de código da figura 5 ilustra o momento em que o agente cliente envia ao agente DP (cujo identificador é representado pela variável `dp`) a SQL referente a sua solicitação de acesso a dados. Neste momento ambos já haviam entrado em um acordo de prestação de serviço.

```
ACLMessage accept;
accept = new
   ACLMessage(ACLMessage.ACCEPT_PROPOSAL);

accept.setReplyWith("DASEDataProvider"
   + System.currentTimeMillis());

accept.addReceiver(dp);
accept.setContent("SELECT * FROM tests WHERE
fielda=5;");

mt = MessageTemplate.MatchInReplyTo(
   accept.getReplyWith());

myAgent.send(accept);
```

Figura 5. Requisição de dados a um agente DP.

7. Resultados

Para avaliar a funcionalidade e o desempenho do sistema, testes foram elaborados. A aplicação de testes consistiu de agentes responsáveis por gerar solicitações de acesso aos dados distribuídos utilizando o DASE. Um SGBD distribuído chamado SisBDPar [24] foi utilizado nos testes. O SisBDPar foi projetado sobre um sistema de arquivos paralelos chamado NPFS [25]. O PostgreSQL, um SGBD centralizado, também foi utilizado para testar a funcionalidade do DASE, comprovando o comportamento esperado.

Para os testes utilizando o SisBDPar, os dados foram distribuídos de forma circular (*round-robin*) em dois nós, em um teste, e em quatro nós em outro teste.

Os resultados dos testes estão apresentados na tabela 1. Em todas as consultas realizadas foram retornadas 10 linhas. Todos os resultados apresentados consideram o tempo de acesso ao SBD através do DASE, e foram executados utilizando computadores com dois processadores Athlon MP 2400+, 1GB de memória e com acesso a discos locais. Nos testes foi utilizado apenas um processador de cada nó.

Tabela 1. Resultados dos testes.

Qtde nós	Tipo de operação	SGBD	Qtde de linhas no SBD	Tempo (ms)
1	SELECT	PostgreSQL	35.000	161
1	SELECT	SisBDPar	35.000.000	3081
2	SELECT	SisBDPar	35.000.000	1541
4	SELECT	SisBDPar	35.000.000	766

Conforme pode ser observado na tabela 1, o acesso aos dados distribuídos utilizando o DASE teve um ganho de desempenho próximo ao número de nós utilizados, tanto para o caso de dois nós, quanto para o de quatro nós. Tal efeito deve-se à busca paralela executada pelo SGBD, permitindo que mais de uma CPU execute simultaneamente instruções de acessos aos dados. Além disso, o tempo de espera durante o acesso ao disco é consideravelmente reduzido, já que há mais de um disco para armazenar os dados paralelamente, e cada um guarda apenas uma parcela do total dos dados.

Através destes testes, também é possível provar a funcionalidade do DASE e exemplificar sua flexibilidade ao suportar diferentes tipos de sistemas gerenciadores de banco de dados.

8. Trabalhos correlatos

De todas as plataformas de agentes pesquisadas [9-12, 16], descritas na terceira seção, nenhuma apresentou qualquer serviço ou funcionalidade, nativa ou acessória, semelhante à proposta pelo DASE, ou mesmo que considerasse o simples acesso a dados centralizados. Assim, certamente estão sujeitas a fazer acesso a dados como qualquer outra aplicação faria, de forma não padronizada, pouco manutenível e flexível, e completamente dependente dos detalhes da requisição de dados, do ambiente de agentes e do SBD utilizado.

9. Conclusões e trabalhos futuros

A necessidade dos agentes por acesso a dados distribuídos é um requisito presente em qualquer sistema multi-agente, já que tais sistemas são normalmente complexos e de natureza distribuída. Este trabalho apresentou alguns detalhes a cerca desta necessidade, além de como supri-la através da proposta de um sistema chamado DASE, *Distributed data Agent Service Environment*.

O DASE é capaz de oferecer a um sistema multi-agente o acesso a dados distribuídos sem que a complexidade do mecanismo de persistência e do manuseio dos dados seja um obstáculo, conferindo transparência e padronização. O DASE também é um sistema multi-agente, portanto oferece sua funcionalidade através de agentes, que são suportados por uma plataforma chamada JADE, *Java Agent DEvelopment Framework*.

As principais melhorias futuras a serem feitas no sistema envolvem o amadurecimento do sistema de controle de concorrência, melhorias no algoritmo de manutenção de agentes provedores de dados, e evolução no balanceamento de carga. Outros aperfeiçoamentos mais complexos envolvem a inclusão de uma camada de persistência de agentes na arquitetura (com um possível mapeamento agente-relacional) e o desenvolvimento de recursos relacionados a controle de acesso e segurança.

Referências

[1] C. Aldo and R. Giovanni, "MULTI-AGENT SYSTEMS AND TERRITORY: Concepts, methods and applications," ERSA CONGRESS, 2003.

[2] E. Oliveira, K. Fischer, and O. Stepankova, "Multi-agent Systems: Which Research for which Applications," Robotics and Autonomous Systems, 1999.

[3] J. R. d. Almeida Junior, L. M. Sato, F. S. Carvalho, I. R. Oliveira, and M. A. Correa, "Modelagem de Dados para Gerenciamento de Tráfego Aéreo," *IV Simpósio de Transporte Aéreo (SITRAER)*, vol. 1, pp. 139-149, 2005.

[4] D. Kinny and M. George, "Modelling and Design of Multi-Agent Systems," presented at Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Australia, 1996.

[5] A. Moreno, AïdaValls, and J. Bocio, "Management of Hospital Teams for Organ Transplants using Multi-Agent Systems," presented at Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine, 2001.

[6] R. Conte, N. Gilbert, and J. S. Sichman, "MAS and Social Simulation: A Suitable Commitment," presented at Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation, 1998.

[7] K. P. Sycara, "Multiagent Systems," in *AI Magazine*, vol. 10, 1998, pp. 79-93.

[8] P. A. Bernstein, "Middleware: A Model for Distributed System Services," *Commun. ACM*, vol. 39, pp. 86-98, 1996.

[9] D. Hiebeler, "The Swarm Simulation System and Individual-based Modeling," *proceedings of Decision Support 2001: Advanced Technology for Natural Resource Management*, 1994.

[10] M. Sonnessa, "JAS: Java Agent-based Simulation library - An open framework for algorithm-intensive simulations" Department of Computer Science, University of Torino.

[11] M. J. North, N. T. Collier, and J. R. Vos, "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit," *ACM Transactions on Modeling and Computer Simulation*, vol. 16, pp. 1-25, 2006.

[12] P. Busetta, R. Rönquist, A. Hodgson, and A. Lucas, "JACK Intelligent Agents - Components for Intelligent Agents in Java," in *AgentLink News*. Melbourne, Australia.: Agent Oriented Software Pty. Ltd., 1999, pp. 2-5.

[13] M. Bratman, *Intention, plans, and practical reason*. Cambridge, MA (USA): Harvard University Press, 1987.

[14] T. Finin and R. Fritzson, "KQML - A Language and Protocol for Knowledge and Information Exchange," presented at Proceedings of the 13th Intl. Distributed Artificial, 1994.

[15] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," in *IEEE Communications Magazine*, vol. 35 No. 2, 1997.

[16] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE - A FIPA2000 Compliant Agent Development Environment," *AGENTS'01*, pp. 216-217, 2001.

[17] M. Aparicio, L. Chiariglione, E. Mamdani, F. MacCabe, R. Nicol, D. Steiner, and H. Suguri, "FIPA - Intelligent agents from theory to practice." Geneva: Telecom 99, 1999.

[18] J. Ellis, L. Ho, and M. Fisher, "JDBC™ 3.0 Specification," 2001.

[19] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, *Patterns: Service-Oriented Architecture and Web Services*: IBM Redbooks, 2004.

[20] L. Rising, "Patterns: a way to reuse expertise," *Communications Magazine*, vol. 37, 1999.

[21] S. Cheung and V. Matena, "Java Transaction API (JTA)," Sun Microsystems Inc., 2002.

[22] P. A. Bernstein and N. Goodman, "Concurrency Control in Distributed Database Systems," *Computing Surveys*, vol. 13, No. 2, 1981.

[23] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, "JADE PROGRAMMER'S GUIDE," 2005, pp. 23-33.

[24] F. A. G. Lubacheski, "Uma Infra-Estrutura para Banco de Dados Baseada em Arquivos Paralelos e Distribuídos," in *Escola Politécnica*, vol. Mestrado. São Paulo: Universidade de São Paulo, 2005.

[25] H. C. Guardia, "Considerações Sobre as Estratégias de um Sistema de Arquivos Paralelos Integrado ao Processamento Distribuído," in *Escola Politécnica*, vol. Doutorado. São Paulo: Universidade de São Paulo, 1999.