Reflexão em Grades Computacionais

Marco A. S. Netto

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

30 de novembro de 2005

Roteiro

Reflexão

Reflexão
Visão Geral

- Início de 80 (Brian C. Smith)
- Sistema observa e modifica sua própria estrutura e/ou comportamento

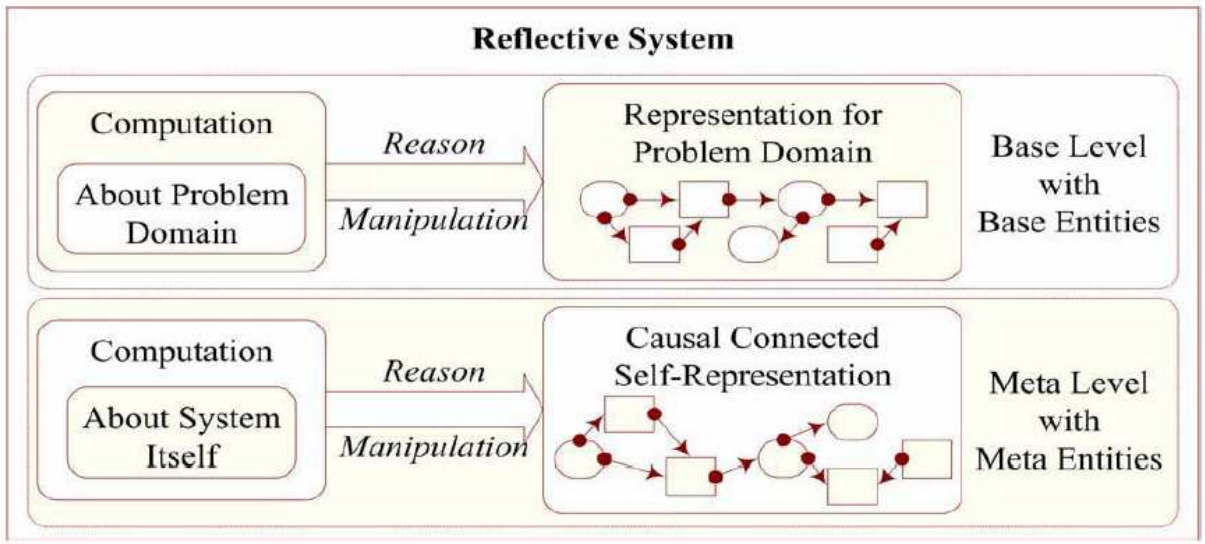
Construção de sistemas:

- Flexíveis
- Adaptáveis
- Reconfiguráveis

Reflexão

Composta em dois níveis:

- Nível base: Computação do problema do usuário (domínio da aplicação)
- Meta nível: sistema reflexivo (meta-domínio) vinculado ao domínio da aplicação



Reflexão
Tipos de Reflexão

- Reflexão Estrutural
  - Obter informação estrutural do programa
  - Possibilitar inspeção, adição, remoção e modificação de características encapsuladas de entidades no nível base
  - Funcionalidades: operações e métodos
  - Estado: variáveis, atributos e constantes
- Reflexão Comportamental
  - Alterar comportamento do programa
  - Interceptar chamadas e valores devolvidos dos métodos

Grades Computacionais

Grades Computacionais

Visão Geral

- Utilização de computadores distribuídos em diversos domínios
- Computação em larga escala
- Ambiente dinâmico e heterogêneo
- Analogia à rede elétrica
- Envolve vários desafios

Grades Computacionais
Desafios

- Heterogeneidade e dinamicidade
- Escalabilidade
- Escalonamento
- Tolerância a falhas
- Segurança

Grades Computacionais
Desafios

- Diversas tecnologias (interoperabilidade)
- Troca de recursos
- Grande quantidade de dados
- Diversos tipos de aplicações
- Qualidade de serviço

Grades Computacionais
Características do Middleware

Sistemas de middleware de Grades devem ser:

- (Re)Configuráveis
- Flexíveis
- Adaptáveis

Reflexão em Grades Computacionais

- Adaptabilidade
- Não específico por aplicação
- Separar requisitos funcionais e não-funcionais
- Busca por separação de interesses
- Simplificação do código
- Aumento do reuso

Reflexão em Grades Computacionais

Projetos

- Legion
- Gridkit
- PAGIS

Reflexão em Grades Computacionais

Projetos - Legion

- Universidade de Virginia (EUA)
- Início em 1993
- Largura de banda
- Primeiro protótipo em 1997
- Applied MetaComputing adquiriu os direitos sobre o Legion em 2001 ($ 16 milhões)
- AVAKI

Reflexão em Grades Computacionais

Projetos - Legion - Reflexão

- Sistema orientado a objetos
- Estruturas internas disponíveis aos desenvolvedores
- Aspectos substituídos por implementações específicas de aplicações
  - Substituir escalonador
  - Módulo de segurança
- Uso principal: personalizar características para diferentes comunidades
- *Reflective Graph and Event* (RGE)

Reflexão em Grades Computacionais

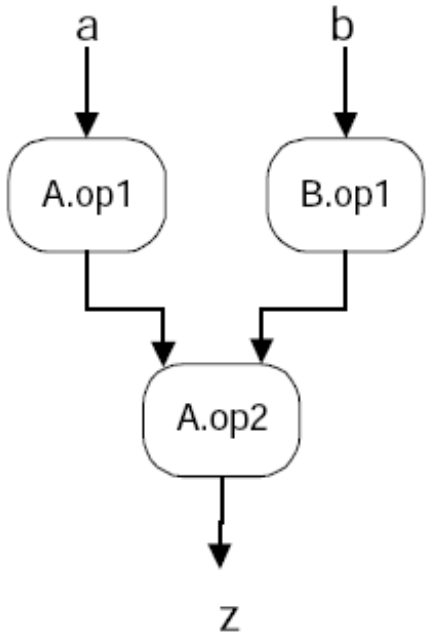
Projetos - Legion - Reflexão - RGE

- Flexibilidade, extensibilidade, reusabilidade, *composability*
- Representar computação através de:
  - eventos, tratadores de eventos e grafos
- Exceções são um tipo de evento

Reflexão em Grades Computacionais

Projetos - Legion - Reflexão - RGE

```
(1) main() {
(2)   int a = 10, b = 15, x, y, z;
(3)   MyObject A, B;
(4)   x = A.op1(a);
(5)   y = B.op1(b);
(6)   z = A.op2(x,y);
(7)   printf("z=%d\n", z);
(8) }
```

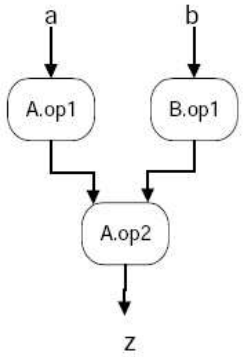


Legion

Code Fragment

```
MyObject A, B;
x = A.op1(a);
y = B.op1(b);
z = A.op2(x,y);
```

Graph Representation



Graph Implementation

```
(1) // Declarations
(2) LegionInvocation inv1, inv2, inv3; // graph nodes
(3) LegionParameter parm;
(4) int a = 10, b = 15, z;

(5) // Object creation
(6) LegionLOID A_name, B_name;
(7) A_name = Legion.CreateObject("MyObject");
(8) B_name = Legion.CreateObject("MyObject");

(9) // Create graph and handles
(10)LegionProgramGraph G(Legion.getMyLoid());
(11)LegionCoreHandle _handle(A_name),B_handle(B_name);

(12)// x = A.op1(a);
(13)inv1 = A_handle.invoke("op1", 1, 1);
(14)G.add_invocation(inv1);
(15)parm = make_parameter(a, 1);
(16)G.add_constant_parameter(inv1, parm,1);

(17)// y = B.op1(b);
(18)inv2 = B_handle.invoke("op1", 1, 1);
(19)G.add_invocation(inv2);
(20)parm = make_parameter(b,1);
(21)G.add_constant_parameter(inv2, parm, 1);

(22)// z = A.op2(a,b);
(23)inv3 = A_handle.invoke("op2", 2, 1);
(24)G.add_invocation(inv1);
(25)G.add_invocation_parameter(inv1, inv3, 1, 1);
(26)G.add_invocation_parameter(inv2, inv3, 1, 2);

(27)G.execute(); // Execute program graph

(28)// Retrieve the return value - print value of z
(29)// <buffer> is a data structure to store arbitrary
    data
(30)buffer = G.get_value(inv3, METHOD_RETURN_VALUE);
(31)buffer.get_int(&z, 1);
(32)printf("z=%d\n", z);
```

Reflexão em Grades Computacionais

Projetos - Gridkit

- Universidade de Lancaster (RU)
- Início em 2004
- Infra-estrutura de comunicação
- Diversos tipos de interações
  - Pedido-resposta com QoS
  - Mensagens confiáveis e não-confiáveis
  - Interações publicar-inscrever
  - Interações par-a-par
  - Comunicação em grupo
  - Workflows

Reflexão em Grades Computacionais

Projetos - Gridkit - Reflexão

- Instância do OpenORB: sistemas dinâmicos e adaptáveis
- Construção de sistemas baseados em:
  - Componentes
  - Arcabouços de componentes
  - Reflexão

Reflexão em Grades Computacionais

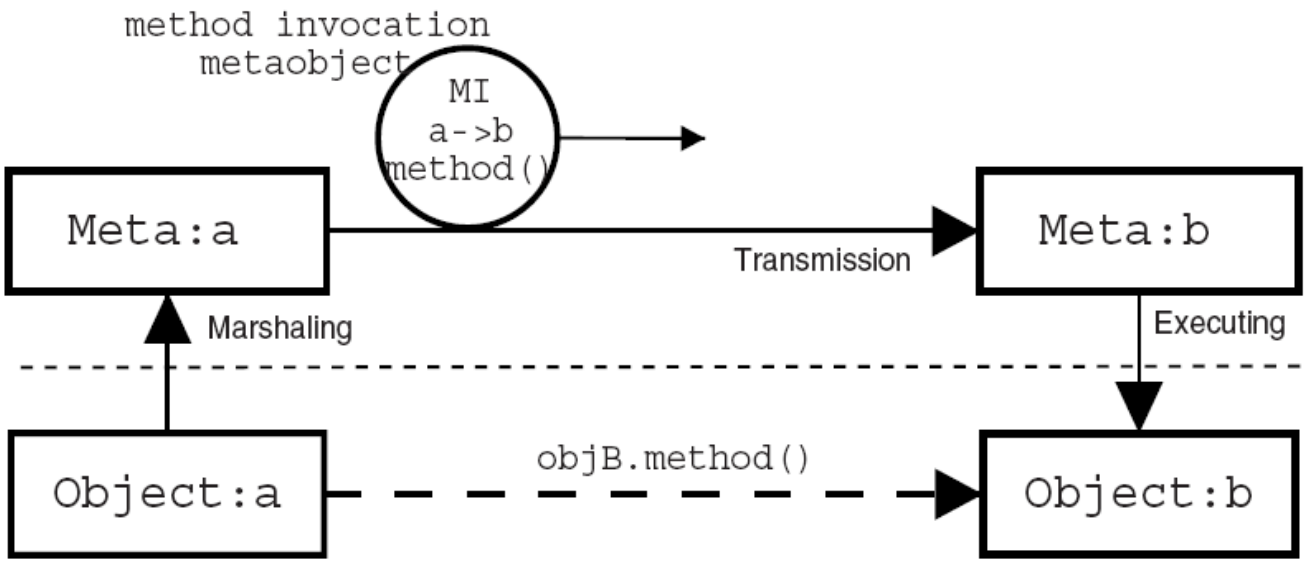
Projetos - PAGIS

- Universidade de Adelaide (Australia)
- Início em 2003
- Desenvolvimento de aplicações para grades
  - Modelos de prog. atuais muito difíceis
- Rede de processos
- Reconfiguração segura das aplicações
- Utiliza Java e o Globus Toolkit

Reflexão em Grades Computacionais

Projetos - PAGIS - Reflexão

- Usuário altera uma das fases: *Marshaling, Transmission, Executing*



Considerações Finais

- Sistemas dinâmicos e heterogêneos
- Complexos
- Crescente expansão
- Flexibilidade
- Reflexão
- Computação autônoma: não há intervenção humana

Principais Referências

- W. Cai et al. **The Gridkit Distributed Resource Management Framework**. European Grid Conference 2005: 786-795 LNCS-Springer.
- A. S. Grimshaw and A. Natrajan. **Legion: Lessons Learned Building a Grid Operating System**. Proceedings of the IEEE Vol 93. No 3. March 2005.
- Anh Nguyen-Tuong et al. **Using Reflection for Flexibility and Extensibility in a Metacomputing Environment**. CS-98-33 (Technical Report - UVa) 1998.
- Darren Webb, Andrew L. Wendelborn. **The PAGIS Grid Application Environment**. International Conference on Computational Science 2003: 1113-1122 (LNCS-Springer).
- G. Coulson et al. **Applying the reflective middleware approach in Grid Computing**. Concurrency and Computation: Practice and Experience 2004; 16:433-440.
- G. Huang et al. **Towards Autonomic Computing Middleware via Reflection**. 28th International Computer Software and Applications Conference 2004: 135-140.

Reflexão em Grades Computacionais

Marco A. S. Netto

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

30 de novembro de 2005