7
Conclusões

Esse trabalho teve como foco principal o estudo dos meios de sincronização paralela disponibilizados em ferramentas de suporte ao desenvolvimento de aplicações paralelas, e realizou o projeto e a implementação do mecanismo de sincronização paralela denominado *Interfaces Coletivas* - ICs.

Essa implementação consistiu da extensão do modelo de componentes SCS com a definição de dois novos conectores especializados em sincronização paralela baseados nas ICs, o *GatherFacet* e o *MulticastReceptacle*, originando o *middleware* de Computação Paralela orientado a componentes SCS-Collective.

O SCS-Collective proporciona alguns aspectos essenciais para o desenvolvimento de aplicações paralelas, como a modularização proporcionada pela abstração de Componentes de Software. Também, em decorrência dessa abstração e das ICs, é facilitada a prática de reuso de software, dado que o comportamento paralelo do componente está definido na interface. Essa abordagem permite a separação de código funcional e código de comunicação [1], característica não presente em ferramentas tradicionais como MPI [6].

Como é baseado no conceito de ICs, o SCS-Collective disponibiliza uma forma simples para configuração das políticas de redistribuição de dados e permite que a paralelização aplicada expresse uma semântica própria da aplicação e manipule diretamente tipos de dados do próprio domínio da aplicação.

A implementação das ICs no SCS-Collective permitiu a identificação dos requisitos básicos para uma implementação do conceito de ICs em middlewares orientados a componentes, além de uma análise dos desafios de uma implementação C++ das ICs. Os requisitos de implementação desse mecanismo e a análise dos desafios de uma implementação C++ se constituem como as principais contribuições desse trabalho.

Testes de desempenho demonstraram a viabilidade da implementação, exibindo resultados de escalabilidade e aceleração próximos de ferramentas bem estabelecidas no cenário de HPC como MPI.

O SCS-Collective possui algumas limitações. Para um suporte ro-

busto à programação de aplicações paralelas se faz necessária a disponibilização de meios de se realizar comunicação MxN como Baude et al. descrevem em [1]. O MPI, por exemplo define as rotinas $all - to - all$ para a implementação da sincronização paralela MxN [6]. O SCS-Collective disponibiliza apenas meios $Mx1$ e $1xN$ de sincronização paralela representados respectivamente pelos conectores *GatherFacet* e *MulticastReceptacle*.

Como o Baude et al. [1] mostram, a interligação de duas interfaces coletivas pode resolver esse problema. Todavia, como não existem ligações diretas entre as entidades do grupo de comunicação, existirá um potencial ponto de gargalo na aplicação.

Também pode-se citar como limitação dessa implementação a não existencia de suporte padrão para a adição de políticas de redistribuição de dados customizadas, assim como é possível na implementação de referência descrita por Caromel [33].

Por fim, alguns trabalhos futuros podem ser elencados:

1. Oferecer uma melhor abordagem para o problema MxN, através da implementação de controladores mais inteligentes para ligações diretas entre as entidades participantes do grupo de comunicação.

2. Geração completamente automática do código de redistribuição, através da implementação do parser de IDL proposto. Essa implementação seria feita na linguagem Lua.

3. Geração automática da definição da própria *sub-interface* interna a partir da definição da *sub-interface* externa e das anotações nela contidas.

4. Suporte a customização de operações de redução assim como no Pro-Active, e criação de um mecanismo geral para a adição de políticas de redistribuição de dados customizadas como *plugins*.

5. Realização de testes de invocação paralela de requisições utilizando a rotina **`send_multiple_requests_deferred`** disponibilizada pela API CORBA DII.

6. Adição de suporte padrão para a realização de operações assíncronas.

7. Adição de suporte a aspectos de tolerância a falhas através, inicialmente, da definição de um *timeout* para cada conector paralelo.

8. Realização de testes em aplicações científicas reais que exercitem a escalabilidade dos algoritmos de redistribuição e validem o mecanismo de geração de código proposto.

9. Análise mais abrangente sobre os padrões de programação paralela suportados pelo mecanismo de *Interfaces Coletivas*.

10. Avaliação de ganho em produtividade e programabilidade no desenvolvimento de aplicações paralelas. Esse estudo poderia ser complementado por uma avaliação do quanto o SCS-Collective consegue separar código de coordenação paralela do código funcional da aplicação.