Extensão da Ferramenta IC2D para Monitoração de Carga em *Clusters* e *Grids* de Computadores

Elton Nicoletti Mathias[1], **Marcelo Veiga Neves**[1], **Edmar Pessoa Araújo Neto**[1], **Marcelo Pasin** [2] **e Andrea Schwertner Charão**[1]

[1]Curso de Ciência da Computação
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brazil

[2]École d'Ingénieurs et d'Architects de Fribourg
Fribourg, Suisse.

{emathias, veiga, araujo, andrea}@inf.ufsm.br, marcelo.pasin@hefr.ch

Abstract. *The analysis of execution environments makes possible a better debugging of distributed programs and to take more accurate decisions about the resources availability for solving tasks. This paper presents an extension to the graphical environment IC2D, that is part of the distributed computing middleware ProActive. The extension developed makes possible the collection and graphical visualization of clusters and grids resources load indexes.*
Key Words : *ProActive, IC2D, Load Index Monitoring, Cluster/Grid Computing*

Resumo. *A análise do ambiente de execução permite uma melhor depuração de programas distribuídos e a tomada de decisões mais apuradas a respeito da disponibilidade de recursos para resolução de tarefas. Esse artigo apresenta uma extensão ao ambiente gráfico de monitoração e controle IC2D, do middleware para computação distribuída ProActive. Essa extensão desenvolvida permite a coleta e visualização gráfica de índices de carga de ambientes de cluster e grid.*
Palavras Chaves : *ProActive, IC2D, Monitoramento de Índices de Carga, Computação para Clusters/Grids.*

1. Introdução

No contexto do desenvolvimento de aplicações distribuídas, sejam elas para execução em *clusters*, ou grades computacionais, a análise do comportamento de aplicações pode ser de grande utilidade para a redução de gargalos, otimização dos algoritmos, ou melhor distribuição de carga. Essas medidas permitem a obtenção de melhores desempenhos, que se refletem em tempos computacionais menores na execução de tarefas. Entretanto, de nada adianta monitorar apenas a aplicação, se seu comportamento sofre influência direta do ambiente onde a mesma está sendo executada.

A inexistência de informações a respeito do ambiente de execução prejudica a depuração de programas, pois nem sempre é possível perceber, por exemplo, que determinada tarefa está levando mais do que o tempo necessário devido a uma alta carga da máquina, ou então que o tempo de uma comunicação é demasiado alto devido ao tráfego

intenso de rede que pode estar ocorrendo na máquina onde a tarefa se encontra. Outra atividade prejudicada pela falta de informações a respeito do ambiente de execução é o lançamento de tarefas, pois não há como saber qual o recurso mais adequado a execução da tarefa.

Para facilitar a depuração de programas distribuídos, o *middleware* ProActive [Caromel et al. 1998] oferece a interface gráfica IC2D [Baude et al. 2001], que permite a visualização e controle de aplicações implementadas a partir dessa biblioteca. Apesar de oferecer uma série de funcionalidades que permitem um controle sobre os objetos monitorados, esta interface não apresenta qualquer tipo de informação a respeito do ambiente onde as aplicações estão sendo executadas, além da topologia onde a aplicação distribui-se.

Como o *middleware* ProActive destina-se à distribuição de tarefas em grades computacionais, torna-se comum a utilização de recursos não dedicados. Nesse tipo de ambiente, a visualização de carga das máquinas pode se tornar ainda mais importante.

Nesse sentido, esse artigo apresenta uma extensão ao ambiente gráfico IC2D, para a geração e visualização de gráficos relativos a índices de cargas de ambientes de execução distribuída. Inicialmente, são apresentadas as ferramentas onde o trabalho inclui-se e uma contextualização do mesmo. Em seguida será apresentada a extensão implementada. Depois, serão apresentados trabalhos relacionados, e uma comparação dos mesmos com o trabalho desenvolvido. Por fim, serão apresentadas algumas considerações finais e trabalhos futuros.

2. ProActive e o Modelo de Objetos Ativos

ProActive [Caromel et al. 1998] é uma biblioteca implementada completamente em Java, que busca oferecer um modelo de programação concorrente e distribuído com transparência. Essa biblioteca é construída inteiramente utilizando a API (*Application Programming Interface*) padrão Java, por isso, ela não requer qualquer modificação no ambiente de execução, uso de compiladores especiais, pré-processadores ou máquina virtual modificada.

Esta biblioteca segue um modelo de programação baseado em objetos ativos. Nesse modelo, cada objeto ativo tem sua própria *thread* de controle, que controla a execução de chamadas de métodos remotos, armazenadas em um sistema de fila.

Os objetos ativos são remotamente acessíveis através da invocação de métodos. As chamadas a esses métodos ocorrem de forma assíncrona e baseiam-se em *objetos futuros*, que são retornados imediatamente após a chamada de método remoto e substituídos automaticamente quando o método retorna realmente. A utilização do valor retornado antes de sua disponibilidade bloqueia o fluxo que chamou o método, em um mecanismo chamado de espera por necessidade.

O *middleware* ProActive também permite a mobilidade de objetos ativos entre diferentes máquinas virtuais Java (JVM), possivelmente localizadas em computadores diferentes. O mecanismo de migração oferecido transfere o objeto ativo para outra JVM, levando consigo o código e seu estado (objetos futuros, chamadas pendentes, etc). O tipo de migração é classificado como migração fraca, por não permitir a migração durante a execução de métodos.

3. A Ferramenta IC2D

Junto ao *middleware* ProActive é disponibilizado o ambiente gráfico IC2D. Este ambiente permite a monitoração e controle de aplicações distribuídas construídas com a biblioteca ProActive. Implementada utilizando RMI e ProActive, a ferramenta trabalha segundo o mecanismo de chamadas assíncronas e migração de tarefas [Baude et al. 2001]. Através de um protocolo de descoberta de recursos próprios permite a monitoração de *máquinas* que não estão disponíveis diretamente.

A versão atual da ferramenta oferece uma série de funcionalidades, organizadas em três módulos distintos:

- Módulo de Visualização Gráfica: Este módulo permite a visualização de *hosts*, JVMs e objetos ativos. Nesse tipo de visualização é possível verificar a topologia na qual a aplicação está distribuída, o estado dos objetos ativos (execução, espera, etc.) e a migração de tarefas;
- Módulo de Visualização Textual: Este módulo proporciona a visualização textual, ordenada, dos eventos gerados no ambiente monitorado. Entre esses eventos podemos citar: lista ordenada de mensagens, dependência causal entre mensagens e eventos relacionados (envio e recepção correspondentes, por ex.);
- Módulo de Controle e Monitoração: Módulo que permite controlar o mapeamento de tarefas através do lançamento das mesmas e a migração interativa de objetos ativos através de um mecanismo "*drag-and-drop*".

Esta ferramenta também apresenta interface com os *middlewares JINI* (Sun) e *Globus*, o que permite a sua utilização como portal para lançamento de aplicações nos ambientes oferecidos por estas ferramentas.

A figura 1 mostra a interface principal da ferramenta IC2D, onde pode ser vista a monitoração de uma aplicação em ambiente distribuído. A janela superior esquerda mostra a janela principal, onde podem ser visualizados objetos ativos no ambiente distribuído, e os eventos gerados no ambiente. A janela superior direita mostra a interface que permite lançamento remoto de aplicações . A janela inferior direita mostra a interface que permite o controle de processos, e à esquerda encontram-se legendas que permitem compreender os componentes gráficos mostrados na janela principal.

4. Extensão da ferramenta IC2D

A fim de permitir, à ferramenta IC2D, oferecer um controle mais efetivo do ambiente de execução, foi implementado um módulo que permite a visualização de gráficos referentes a índices de carga, coletados nas máquinas monitoradas (ocupação do processador e memória, tráfego de rede, carga média, etc).

A extensão implementada permite visualizar informações históricas de monitoração em *grids*, *clusters* e *hosts*, a partir do armazenamento cíclico dos índices coletados. Essa visualização permite a verificação das tendências de ocupação existentes nas amostras coletadas em diferentes granularidades de tempo, em intervalos que vão desde minutos a anos.

A subseções a seguir mostram como funciona a coleta e armazenamento das métricas monitoradas, o protocolo de comunicação utilizado e a interface onde os gráficos referentes aos índices coletados são exibidos.

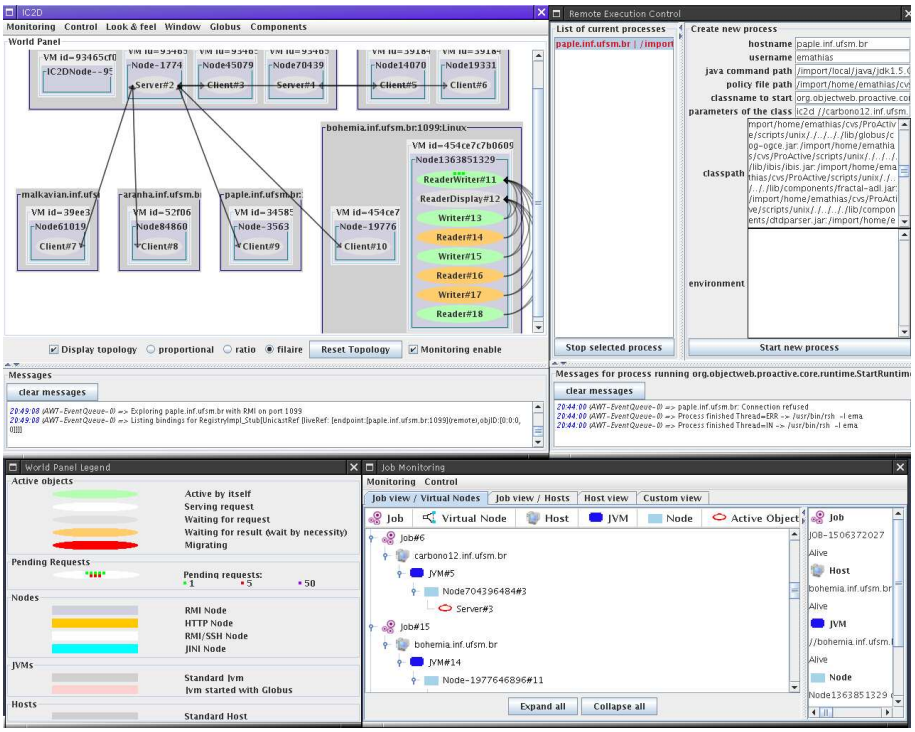

Figure 1. Interface principal da ferramenta IC2D

4.1. Coleta de Métricas

Por ser implementádo todo em Java, o *middleware* ProActive é portável a vários sistemas operacionais e arquiteturas, dependendo apenas da existência de uma JVM compatível. Entretanto, a coleta de métricas é realizada de formas diferentes em cada um dos sistemas operacionais.

Para contornar essa dificuldade, foi implementada uma biblioteca em linguagem nativa (C) que permite a coleta de dados em diversos sistemas operacional, para diversas arquiteturas. Atualmente os sistemas operacionais suportados são: Linux (i386, ia64, sparc, alpha, powerpc, m68k, mips, arm, hppa, s390), Solaris, FreeBSD, AIX, IRIX, Tru64, HPUX, MacOS X e Windows NT/XP/2000. Para reduzir a intrusividade, esta biblioteca também permite a coleta junto a algumas ferramentas de monitoração de *clusters* e *grids*, que podem já estar efetuando a coleta de dados. Atualmente, as ferramentas suportadas são: Ganglia, Performance Co-Pilot, Parmon e SCMS. Também é possível a coleta através de dados via SNMP. Além de permitir a manutenção da portabilidade, a utilização de métodos nativos também permite uma minimização da intrusividade, já que a leitura em código nativo apresenta menor custo computacional. A ligação entre código nativo e Java é feita por uma interface JNI (*Java Native Interface*).

Para realizar a coleta, cada máquina possui um *daemon* responsável pelo registro periódica dos dados, que são armazenados em uma base de dados circular, de tamanho fixo. O intervalo escolhido para a coleta das métricas é, por padrão, de 10 segundos. Este intervalo foi determinado para evitar o aumento da sobrecarga dada pela coleta e pela necessidade de comunicação dos dados à máquina onde o usuário monitora o ambiente. Pode-se configurar intervalos menores, mas dependendo do hardware monitorado e da quantidade de máquinas monitoradas, a operação de coleta e comunicação dos valores

coletados pode causar sobrecarga no ambiente.

Os índices coletados dividem-se em métricas de valor constante, que são coletadas apenas uma vez, e métricas que tem seu valor alterado com o decorrer do tempo, as quais necessitam de coleta cíclica. Entre as métricas constantes tem-se o número e freqüência de processadores, tamanho dos discos rígidos e quantidade de memória principal e *swap*. As métricas dinâmicas monitoradas são utilização de processador, memória principal, *swap* e disco, carga média nos últimos 1, 5 e 15 minutos e o número de bytes e pacotes entrando e saindo pelas interfaces de rede.

Além da coleta para visualização, o coletor também disponibiliza uma API pública, de acesso às métricas coletadas. Essa interface permite o acesso simplificado a características de hardware, ou carga, podendo beneficiar aplicações que queiram fazer uso desse tipo de informação.

4.2. Armazenamento

Para realizar o armazenamento dos dados, há a necessidade de manter histórico das amostras coletadas. A extensão implementada utiliza a biblioteca JRobin, Trata-se de uma biblioteca Java, de código aberto, que permite a manipulação de arquivos RRD (*Round Robin Database*). Esses arquivos constituem-se em bases de dados circulares com tamanho fixo. Cada RRD organiza-se em tabelas chamadas RRAs (*Round Robin Archives*). Depois que cada tabela alcança seu tamanho máximo, os novos dados inseridos sobrescrevem os valores mais antigos. Apesar de sobrescritos, estes valores não são, de todo, perdidos, pois é aplicado sobre eles uma função de consolidação (no caso deste trabalho, a média aritmética). Os dados consolidados são armazenados em outro RRA, do mesmo RRD.

O *daemon* coletor cria um RRD para cada métrica variável coletada. Cada RRD é composto por 6 RRAs, que se destinam ao armazenamento de métricas referentes aos últimos minutos, horas, dias, semanas, meses e anos. Embora, para a construção dos gráficos, o armazenamento seja necessário apenas no computador onde está sendo executada a interface gráfica IC2D, o armazenamento local continua sendo necessário pois, dessa forma, é possível a visualização de gráficos referentes a períodos anteriores à execução da interface. A seção seguinte descreve o protocolo utilizado para comunicação dos dados entre os coletores e a interface gráfica.

4.3. Protocolo de Comunicação

A comunicação entre os módulos coletores, presentes em cada nó, e a interface gráfica ocorre em dois momentos. Quando se inicia a monitoração e ciclicamente após a comunicação inicial.

Quando incluída na interface de monitoração, cada máquina converte a base de dados para um arquivo XML (*Extended Markup Language*). O arquivo gerado é enviado à máquina onde reside a interface gráfica, convertido novamente ao formato RRD e salvo em local apropriado. Nesse momento ocorre a sincronização de tempo entre a base de dados recebida e o horário local. Como a menor unidade de tempo permitida em uma base RRD é de 1 segundo e a visualização mais fina da interface é da ordem de minutos, o protocolo de sincronização utilizado é bastante simples, consistindo apenas de uma chamada

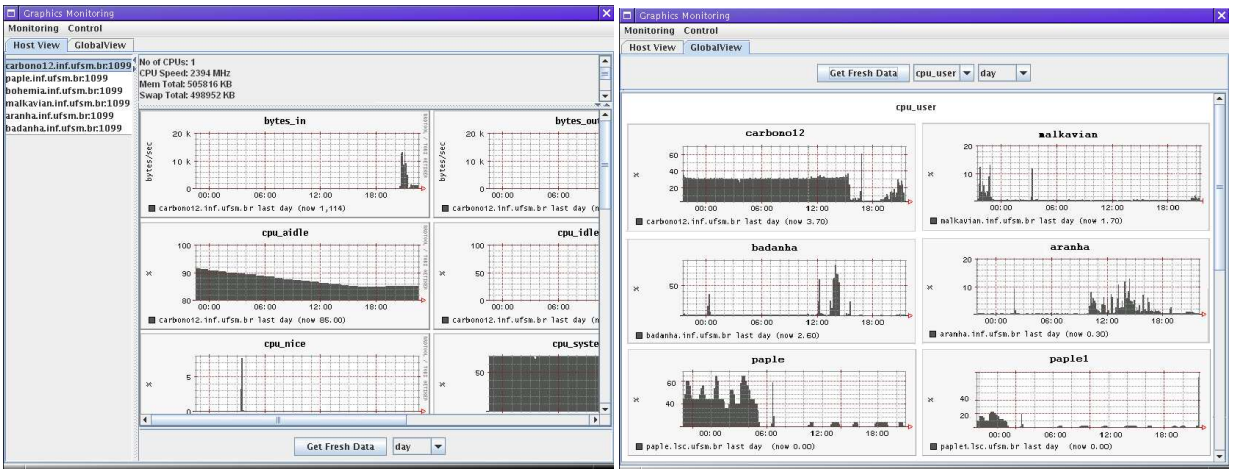

Figure 2. (a)Visualização "*Host View*" (b)Visualização "*Global View*"

de método na máquina monitorada, que retorna o tempo local à máquina, descontado da metade do tempo que levou a chamada.

Depois de incluído na interface, cada nó envia, ciclicamente, as métricas coletadas à máquina onde está ocorrendo a monitoração. O intervalo de envio é definido segundo o TTR (*Time To Refresh*) definido pelo usuário na interface. Cada vez que ocorre modificação no TTR, todos os nós monitorados são notificados. Para minimizar a ocorrência de gargalos que podem existir em decorrência da tentativa de todos os nós monitorados tentarem enviar seus dados coletados no mesmo instante, cada nó monitorado envia seu último valor coletado, em um tempo escolhido randomicamente dentro do intervalo determinado pelo TTR.

4.3.1. Visualização

Depois de disponíveis localmente, os arquivos das métricas coletadas em cada um dos nós monitorados, e de suas atualizações acionadas remotamente, os gráficos podem ser gerados a partir dos arquivos RRD. A geração de gráficos ocorre através da mesma ferramenta que proporciona a manipulação de arquivos RRD (JRobin), que agrega em seu pacote a ferramenta FreeJChart, para a geração de dados referentes às séries temporais armazenadas.

A interface desenvolvida permite dois tipos diferentes de visualização, em abas separadas:

- *Host View* (Figura 2.a): Permite a visualização de informações referentes a um *host*, escolhido entre os monitorados. As métricas estáticas são mostradas separadamente das dinâmicas, as quais podem ter sua granularidade definida em minutos, horas, dias, semanas, meses ou anos. Esta visualização apresenta todas as métricas coletadas, referentes a este *host*.
- *Global View* (Figura 2.b): Permite a visualização de determinada métrica simultaneamente em todos os hosts monitorados, segundo as mesmas granularidades da visualização anterior.

Além da visualização, a interface também apresenta funcionalidades referentes a inserção ou remoção de nós na monitoração e outras configurações como TTR.

5. Trabalhos Relacionados

A área de monitoração e visualização de aplicações e sistemas distribuídos é bastante ampla. Existem muitas ferramentas que buscam realizar essas tarefas, com diferentes enfoques. Algumas ferramentas focalizam na monitoração do ambiente a fim de oferecer, principalmente a administradores mecanismos de avaliar o estado dos recursos computacionais. Outras focalizam a monitoração de aplicações, voltadas principalmente a desenvolvedores na tarefa de depuração de programas distribuídos. A versão original da ferramenta IC2D tem o enfoque do segundo grupo de ferramentas. Entretanto, a partir da extensão desenvolvida, características do primeiro grupo passam, também, a estar presentes.

Dentre as ferramentas que focalizam a monitoração de ambientes (*grids* e *clusters*) podemos citar o Ganglia [Massie et al. 2003], SCMS [Uthayopas and Rungsawang 1999], e Parmon [Buyya 2000], entre outras. Estas ferramentas são bastante distintas. Ganglia volta-se à arquiteturas hierárquicas, organizadas em federações de *clusters*, e apresenta gráficos relativos às métricas coletadas em uma interface Web. SCMS volta-se a clusters de pequeno e médio porte, apresentando funcionalidades como a execução de comandos em paralelo. Parmon, por sua vez, volta-se à monitoração de clusters, permite a execução de comandos em paralelo e geração de alarmes condicionados a determinadas condições do sistema.

Apesar de não possuir interface Web como Ganglia, nem permitir a execução de comandos paralelos, como o SCMS e Parmon, o IC2D, com a extensão desenvolvida, apresenta algumas vantagens. A primeira delas é a de permitir a visualização de gráficos referentes a diferentes domínios administrativos, sem necessidade de instalação de ferramentas adicionais nesses domínios. Além disso, permite aproveitar ferramentas de monitoração já instaladas, reduzindo a sobrecarga.

Na outra classe, das ferramentas que focalizam a monitoração de aplicações, podemos citar as ferramentas XPVM [Geist et al. 1994] e ParaGraph [Ries et al. 1993]. XPVM permite a depuração de aplicações que utilizam a biblioteca PVM, em *clusters*. Paragraph constitui-se em um ambiente gráfico oferecido no ambiente de monitoração *Paragon*, e apresenta animações do tráfico de mensagem ou atividade dos nós processadores, entretanto permite apenas análise post-mortem .

Nenhuma das ferramentas voltadas ao monitoração de aplicações citadas acima apresenta geração de gráficos referentes a carga de máquinas, nas quais aplicações estão sendo monitoradas. Além disso, voltam-se apenas ao ambiente ao qual destinam-se, enquanto o IC2D permite interface com outras ferramentas, como Globus, JINI e Ibis [Baude et al. 2001].

Além das características apresentadas anteriormente, outra contribuição deste trabalho é oferecer uma API pública, que pode servir a aplicações que necessitem dados referentes ao sistema em que estão rodando.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma extensão da ferramenta IC2D, que permite a visualização de gráficos referentes a índices de carga. A extensão implementada mantém a portabilidade do *middleware* ProActive, no qual se integra. Através da utilização de métodos nativos e

bases de dados circulares , permite uma coleta de dados e transmissão de dados de forma a minimizar a intrusividade.

A visualização de gráficos de utilização dos nós monitorados complementa as funcionalidades disponibilizadas na versão original da ferramenta, pois permite ao usuário um controle mais efetivo da criação, lançamento e migração de tarefas, já que possibilita a visualização de características de cargas atuais, ou históricas para tomada dessas decisões. A título de exemplo, com a visualização oferecida por esse novo módulo, torna-se possível o lançamento de tarefas em máquinas menos carregadas ou então a migração de objetos que estão trafegando um grande número de pacotes, entre si, para a mesma máquina.

A versão atual ainda não proporciona a monitoração de máquinas inacessíveis diretamente da máquina de onde está ocorrendo a monitoração, como as demais funcionalidades da ferramenta IC2D, mas pretende-se que esta funcionalidade também seja oferecida em breve. Pretende-se também a realização de testes de escalabilidade, para verificar o comportamento da extensão implementada frente a um número grande de máquinas monitoradas, em locais distantes geograficamente.

References

Baude, F., Bergel, A., Caromel, D., Huet, F., Nano, O., and Vayssière, J. (2001). Ic2d: Interactive control and debugging of distribution. In *Proceedings of the Third International Conference, LSSC 2001*, volume 2179 of *LNCS*, pages 193–200, Sozopol, Bulgaria. Springer-Verlag.

Buyya, R. (2000). PARMON: a portable and scalable monitoring system for clusters. *Software Practice and Experience*, 30(7):723–739.

Caromel, D., Klauser, W., and Vayssière, J. (1998). Towards seamless computing and metacomputing in Java. *Concurrency: Practice and Experience*, 10(11–13):1043–1061.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994). *PVM Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, Mass.

Massie, M., Chun, B., and Culler, D. (2003). The ganglia distributed monitoring system: Design, implementation, and experience. Technical report, University of California, Berkeley Technical Report.

Ries, B., Anderson, R., Auld, W., Breazeal, D., Callaghan, K., Richards, E., and Smith, W. (1993). The paragon performance monitoring environment. In *Supercomputing '93: Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, pages 850–859, New York, NY, USA. ACM Press.

Uthayopas, P. and Rungsawang, A. (1999). SCMS: An extensible cluster management tool for beowulf cluster. In *Proceedings of Supercomputing'99 (CD-ROM)*, Portland, OR. ACM SIGARCH and IEEE.