Uma ferramenta de monitoramento hierárquico para um sistema maciçamente paralelo virtual

Ana Maria de Seixas Pereira
Centro Nacional de Processamento de Alto Desempenho em São Paulo - CENAPAD-SP
Universidade Estadual de Campinas - UNICAMP
Campinas, SP, Brasil
Email: ana@cenapad.unicamp.br

Marco Aurélio Amaral Henriques
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas - UNICAMP
Campinas, SP, Brasil
Email: marco@dca.fee.unicamp.br

***Resumo*—Apresentamos neste artigo a implementação e a avaliação de JoiNMon, uma ferramenta de monitoramento para o sistema JoiN - ambiente virtual para processamento maciçamente paralelo. Tentamos incorporar nesta ferramenta as melhores características encontradas em algumas das ferramentas de monitoramento disponíveis para ambientes distribuídos. Buscamos também apresentar soluções para as principais dificuldades encontradas na implementação deste tipo de ferramenta. JoiNMon realiza a captura e a publicação de um grande número de informações, as quais permitem uma observação mais detalhada do sistema JoiN, contribuindo com sua administração e uso. Os resultados mostram que a ferramenta oferece uma boa relação custo/benefício no monitoramento deste tipo de sistema, sem causar impactos significativos na execução de aplicações paralelas.**

Palavras-chave: sistema maciçamente paralelo, sistema paralelo virtual, ferramenta de monitoramento

I. Introdução

O sistema JoiN permite a exploração de ciclos ociosos de computadores conectados a Internet, disponibilizando poder computacional para aplicações que possam tirar proveito de um grande paralelismo de dados [1], [2].

Os computadores conectados ao sistema JoiN pertencem a colaboradores, que colocam seus computadores a serviço do sistema e contribuem com ciclos de cpu, constituindo ambientes heterogêneos e maciçamente paralelos de "computação voluntária", também chamados de ambientes de Computação em Grade com Recursos Públicos (*PRC - Public-Resource Computing*) [3]. Nestes ambientes, a utilização do tempo ocioso de computadores quaisquer viabiliza a execução de aplicações paralelas que requerem computação de alto desempenho e que podem se adequar a uma infra-estrutura fracamente acoplada.

Avaliamos a possibilidade de implentação no sistema JoiN de alguma das ferramentas de monitoramento utilizadas em ambientes de computação em grade ou distribuídos, como por exemplo: Ganglia [4], MonALISA [5], Lemon [6], JAMM [7] e R-GMA [8]. Entretanto, devido à arquitetura particular do sistema JoiN e às suas características específicas, concluiu-se que nenhuma destas poderia ser utilizada diretamente pelo JoiN, tornando-se necessário o desenvolvimento de uma ferramenta específica para este ambiente, aproveitando os pontos fortes identificados nas ferramentas analisadas.

A ferramenta JoiNMon foi implementada com base na arquitetura GMA (*Grid Monitoring Architecture*) [9], que propõe padrões e interoperabilidade entre os *softwares* desenvolvidos para monitoramento em *Grid Computing*, com o objetivo de facilitar o monitoramento e a administração do sistema JoiN. Na sua implementação buscamos soluções eficientes para a obtenção e publicação de informações de forma a interferir o mínimo possível no desempenho do sistema e tais soluções podem ser aplicáveis e úteis também em outros sistemas.

O artigo é organizado da seguinte forma: a seção II apresenta o sistema JoiN e as características necessárias em uma ferramenta para seu monitoramento; na seção III apresentamos as etapas de implementação de JoiNMon; a seção IV apresenta a avaliação do desempenho da ferramenta, a análise da sobrecarga que introduz na execução de aplicações e no tráfego de rede, avaliando alterações no desempenho do sistema; na seção V apresentamos as conclusões e sugestões para trabalhos futuros.

II. Sistema JoiN

JoiN é um sistema que tem por objetivo permitir a criação e utilização de ambientes de grade computacionais com recursos públicos, e com isto, viabilizar a execução de aplicações maciçamente paralelas [1].

Desenvolvido em linguagem Java, com uma estrutura baseada em componentes, o sistema JoiN tem como principal característica a grande portabilidade, pois pode ser executado em qualquer computador em que houver uma implementação da Máquina Virtual Java (JVM). A escalabilidade, essencial para ambientes de grade, é obtida pela organização hierárquica dos computadores em grupos. Além disto, JoiN implementa eficientemente funções importantes para este tipo de sistemas tais como escalonamento, replicação, tolerância a falhas e segurança [2].

A implementação do sistema JoiN é baseada no conceito de serviços, que podem ser combinados de diversas formas para compor ambientes bastante flexíveis. Sua arquitetura, como pode ser visto na Figura 1, é baseada em componentes hierarquicamente organizados: um mediador (*server*), módulos de administração (*jack*) e grupos interconectados. Cada um destes grupos é formado por um *coordinator* e diversos *workers*. Os

workers são responsáveis pela execução das aplicações e o número deste tipo de componente pode ser muito grande.

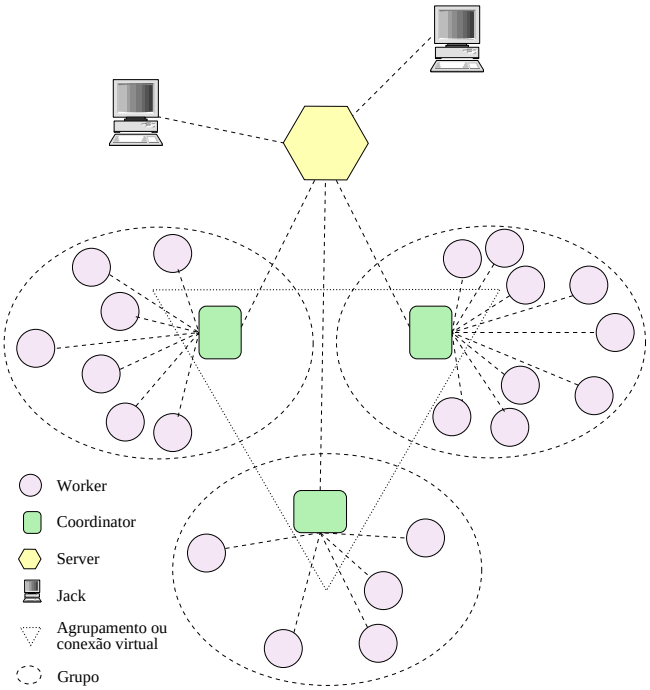


Figura 1. Arquitetura do sistema JoiN

As aplicações são formadas de muitas tarefas (*tasks*) paralelas que são distribuídas pelos *workers*. Os *coordinators* têm como objetivo fazer uma distribuição adequada de tarefas pelos *workers*, equilibrando carga, bem como coletar os resultados das mesmas. Na arquitetura hierárquica do sistema JoiN, cada *worker* só conhece seu respectivo *coordinator* e cada *coordinator* conhece os demais *coordinators* e o mediador (*server*). Isto provê uma maior escalabilidade para o sistema, bem como viabiliza a distribuição e execução de uma grande aplicação por vários grupos. O mediador (*server*) atua como ponto de contato inicial para todos os computadores que se conectam, ou reconectam, ao sistema oferecendo seus serviços. Após o contato inicial, tais computadores são designados como *coordinators* ou são encaminhados a um *coordinator* já existente para atuarem como *workers* do mesmo. Finalmente, o módulo de administração (*JACK - Join Administration and Configuration Kit*) se propõe a servir de interface entre o sistema e seus administradores/usuários. Ele provê ferramentas para instalar, executar e parar aplicações paralelas no Join. Mais detalhes sobre esta plataforma e comparações com outros sistemas são feitas nas referências já citadas.

Neste tipo de sistema é de fundamental importância coletar informações que permitam conhecer o comportamento das aplicações e da própria plataforma de processamento paralelo. Só assim será possível identificar problemas e gargalos a serem removidos para otimizar o uso dos recursos disponíveis. Por outro lado, as características específicas da forma como Join foi construído dificultam sua integração com sistemas de monitoramento já existentes, pois estima-se que o esforço de adaptação seria muito grande e a interferência do monitoramento na plataforma poderia ser significativa.

As características de um ambiente que utiliza o sistema *JoiN* são: maciçamente paralelo, porque pode ser composto por muitos computadores geograficamente distribuídos; heterogêneo, porque agrega computadores de diferentes arquiteturas e sistemas operacionais administrados de forma independente por seus proprietários; dinâmico, porque os computadores que o compõem podem conectar-se e desconectar-se a qualquer momento. Tais características dificultam seu monitoramento e gerenciamento, já que envolve o tratamento de um grande volume de informações em um ambiente pouco controlado.

A ferramenta de monitoramento deve prover informações claras e atualizadas sobre aplicações, usuários, colaboradores e o estado do sistema de uma forma geral. Para isto deve coletar e tornar disponíveis informações sobre: os componentes do sistema (*server*, *coordinators* e *workers*); os colaboradores do sistema e seus computadores; os usuários do sistema, suas aplicações e tarefas.

Além disto, a ferramenta deve observar as restrições impostas pela arquitetura de serviços do sistema. No caso do JoiN, estas restrições têm por objetivo manter a escalabilidade e a portabilidade, essenciais em ambientes heterogêneos e dinâmicos de computação em grade [1].

III. A FERRAMENTA JOINMON

A padronização proposta na arquitetura GMA consiste de um modelo produtor-consumidor que pôde ser adaptado à arquitetura do sistema JoiN. Como GMA não impõe protocolos ou modelo de dados específicos, também foi possível buscar um modelo de dados mais adequado às informações que se deseja monitorar. Além disso, a comunicação entre os componentes da ferramenta pôde se adequar às restrições impostas pela arquitetura do sistema JoiN.

As informações obtidas, para cada computador conectado e para cada aplicação em execução no sistema JoiN, são processadas pela ferramenta de monitoramento JoiNMon e exibidas por meio de uma interface gráfica, responsável pela interação entre o serviço de coleta de dados e os usuários do sistema, obtendo e apresentando adequadamente as informações solicitadas. As informações coletadas são também gravadas em um banco de dados relacional, para que possam ser recuperadas e analisadas posteriormente.

No planejamento e na implementação de JoiNMon foram observadas as seguintes questões:

1) identificação das informações relevantes para o monitoramento, que reflitam o estado atual do sistema e aplicações em execução, bem como informações sobre estados passados e aplicações já encerradas;
2) avaliação da melhor forma de obtenção e transmissão destas informações;
3) definição do armazenamento destas informações de forma eficiente para facilitar sua recuperação;
4) definição de uma interface gráfica onde possam ser obtidas facilmente as informações desejadas.

Na arquitetura GMA são definidos três componentes: produtores, consumidores e serviços de diretório. Os sensores geram

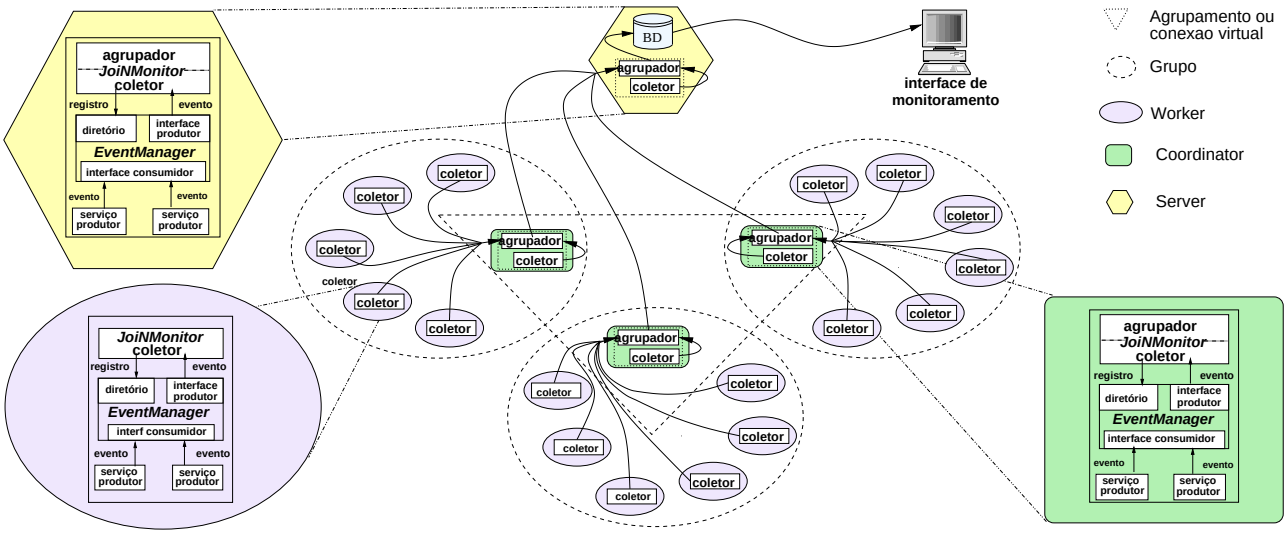


Figura 2. JoiNMonitor - arquitetura e hierarquia

as informações necessárias e as enviam aos componentes produtores que as coletam, tratam e disponibilizam de forma a permitir o monitoramento das informações relevantes pelos componentes consumidores.

Esta arquitetura reúne as características necessárias para monitoramento de recursos do sistema JoiN, pois os componentes do modelo produtor/consumidor proposto podem ser implementados como serviços e executados, conforme a hierarquia da arquitetura JoiN, nos componentes *worker*, *coordinator* e *server*.

As principais informações coletadas são:

- Aplicações: identificação, usuário, início e término do processamento, computadores utilizados;
- Usuários: identificação, acessos ao sistema, aplicações executadas, tempo de processamento utilizado;
- Computadores (componentes *worker*): identificação, características (S.O., Java), aplicações em execução;
- Colaboradores: identificação, computadores, tempo de processamento, tempo das conexões;
- Componentes *server* e *coordinator*: identificação, caracteristicas (S.O., Java), aplicações em execução, componentes *worker* conectados.

A ferramenta JoiNMon foi desenvolvida em dois módulos: o primeiro é o módulo para coleta, tratamento e armazenamento das informações, implementado como um serviço do sistema JoiN; o segundo módulo implementa uma interface gráfica que possibilita a análise das informações previamente armazenadas e, desenvolvida em JSP (Java Server Pages), pode ser utilizada com navegadores *web*. A composição da ferramenta de monitoramento JoiNMon pode ser apresentada como:

- produtores de eventos (diversos serviços do sistema JoiN);
- serviço para gerenciamento de eventos (*EventManager*);
- serviço para consumo de eventos (*JoiNMonitor*);
- interface gráfica (aplicação web *JoiNMonitor*).

A produção de eventos é feita por serviços já existentes no sistema. Estes serviços são responsáveis pela coleta das informações e por sua transmissão, por meio de eventos. Desta forma, o monitoramento em cada nó é feito pela coleta das informações desejadas e a geração de eventos para que as mesmas sejam adequadamente tratadas.

Para o tratamento de eventos é utilizado o serviço *eventManager* do JoiN. Os eventos são tratados de forma assíncrona: a geração de um evento corresponde à sua inclusão em uma fila; uma *thread* especializada obtém eventos desta fila e executa os consumidores correspondentes. Assim, o serviço que produz um evento não tem que aguardar a execução dos consumidores deste evento, e não fica bloqueado.

O consumo de eventos é feito pelo serviço *JoiNMonitor* implementado no sistema JoiN por meio de dois elementos (Figura 2), respeitando a arquitetura hierárquica do sistema:

- coletor: executado em todos os componentes (*worker, coordinator* e *server*), é responsável pelo tratamento das informações contidas nos eventos gerados pelos demais serviços;
- agrupador: executado nos componentes *coordinator* e *server*, é responsável pelo agrupamento das informações, seguindo a hierarquia de componentes do sistema JoiN. No componente *server* também é feito o arquivamento das informações, utilizando para isto uma base de dados relacional.

As informações coletadas pela ferramenta de monitoramento são armazenadas em uma base de dados, mantida no componente *server*. Foi utilizada uma base de dados relacional, onde os dados são mantidos em tabelas projetadas para otimizar os acessos, tanto para escrita quanto para leitura. Na primeira versão foi utilizado o servidor de banco de dados HSQLDB [10] devido à sua simplicidade e desempenho.

A interface gráfica, desenvolvida em JSP, acessa as informações na base de dados mantida pelo serviço JoiNMonitor e as exibe de forma organizada. O *layout* das páginas foi projetado para facilitar a navegação e a obtenção de informações.

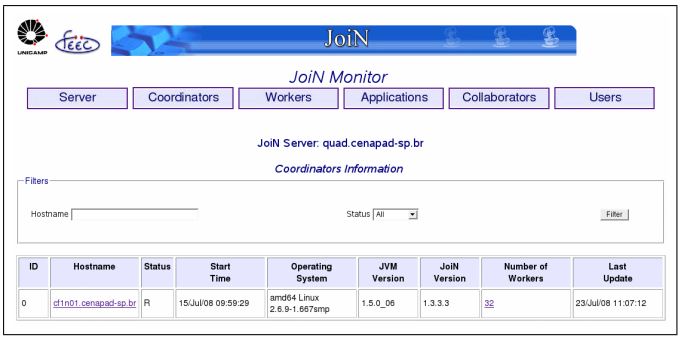


Figura 3. JoiNMonitor - informações componentes coordinator

Na página inicial são exibidas as informações do componente *server* e, através de botões localizados na parte superior da tela é possível navegar entre as telas que contêm informações sobre os demais componentes (*coordinators* e *workers*), e sobre as aplicações, colaboradores e usuários.

Logo abaixo destes botões de navegação está uma área para seleção e filtro das informações a serem exibidas, seguida da área onde são apresentadas as informações. Algumas destas informações são *links* para outras páginas que trazem informações mais detalhadas.

Um exemplo desta interface pode ser visto na Figura 3, onde são apresentadas informações de componentes *coordinator*.

IV. Avaliação de resultados

A avaliação do impacto da ferramenta JoiNMon sobre o sistema JoiN e seus usuários foi feita por meio da análise da sua funcionalidade e da sua eficiência, atentando para as vantagens e possíveis desvantagens em sua adoção. Na análise de sua funcionalidade concluímos que a ferramenta JoiNMon atende as expectativas de prover, a um baixo custo computacional, informações básicas fundamentais para a compreensão do funcionamento do sistema.

A avaliação da eficiência da ferramenta JoiNMon foi feita pela medição da sobrecarga que a mesma introduz no sistema JoiN, no tempo de execução das aplicações e no tráfego de rede. Esta avaliação foi feita por meio de experimentos, apresentados e discutidos a seguir.

A. *Eficiência - realização de experimentos*

Foram feitos experimentos com diferentes configurações do sistema, variando o número de componentes *worker*, o tipo de aplicação e o número de tarefas. Para execução destes experimentos foi utilizado um ambiente homogêneo, composto por um *cluster* com 24 nós, cada um dos nós com a seguinte configuração: 2 processadores AMD-Opteron 242, 1.6GHz e 2GB RAM; interconexão por meio de Gigabit Ethernet; JoiN Version 1.3.3.3; Operating System Linux Fedora 3 (2.6.9-1.667smp); JVM Version 1.5.0_06.

Cada uma das aplicações foi executada em diferentes configurações e, para cada configuração, os experimentos foram feitos em duas etapas: na primeira etapa foi utilizado o sistema JoiN sem a ferramenta de monitoramento; na segunda etapa a ferramenta de monitoramento foi ativada e os experimentos repetidos.

Além disto, cada experimento foi repetido pelo menos cinco vezes, para que pudesse ser calculada a média dos valores obtidos, que é o valor utilizado nas avaliações.

Foi utilizado um computador, com a mesma configuração e ligado na mesma rede, para a execução dos componentes *server* e *coordinator*. Foram realizados experimentos com 4, 8, 12, 16, 24 e 32 componentes *worker*.

Para as análises de tráfego de rede foram utilizadas informações capturadas pela ferramenta Wireshark Version 1.0.0 [11], que é uma ferramenta para análise de protocolos de rede, que facilita a seleção dos dados, capturados por meio do comando *dumpcap* existente em sistemas *Unix*.

Em JoiN as aplicações são normalmente classificadas em três grupos conforme a razão entre o tempo consumido em computação e o tempo consumido para a comunicação necessária para sua execução. Definimos os seguintes grupos, onde R_{cc} = *tempo_de_computação / tempo_de_comunicação*:

1) $R_{cc} \geq 100$: onde o tempo de computação é muito maior que o tempo consumido para comunicação. Nesta classe estão as aplicações que melhor utilizam os recursos do sistema JoiN, pois podem explorar mais o paralelismo.
2) $1 < R_{cc} < 100$: onde o tempo de computação ainda é superior ao tempo de comunicação, mas não muito. As aplicações desta classe também exploram o paralelismo disponível, mas não tão bem como no caso anterior e, assim, não se consegue uma alta eficiência; caso mais comum quando há uso de redes heterogêneas e compartilhadas.
3) $R_{cc} \leq 1$: onde o tempo de computação é menor ou igual ao tempo gasto em comunicação. As aplicações desta classe não são adequadas para o uso de ambientes paralelos e não são consideradas.

Com base nesta classificação selecionamos duas aplicações diferentes para que os experimentos permitissem a observação e a análise do comportamento da ferramenta JoiNMon nas distintas condições em que o sistema JoiN pode ser utilizado. Utilizamos uma aplicação que faz o cálculo do número π pelo método Monte Carlo ($R_{cc} \geq 100$) e uma aplicação para multiplicação de matrizes ($1 < R_{cc} < 100$).

As aplicações escolhidas poderiam fazer parte de benchmarks padrão, como o *NAS Parallel Benchmark* [12] por exemplo, mas nesta fase inicial de análise, optamos por focar em aplicações que tivessem um comportamento bem previsível a fim de melhor avaliar problemas iniciais com a plataforma sob teste. O cálculo de π por Monte Carlo foi escolhido por apresentar um R_{cc} superior a 450. Ele se enquadra dentro do grupo de aplicações consideradas *"embarrassingly parallel"*, que são uma parte dos benchmarks mais modernos. Já a multiplicação de matrizes tem um R_{cc} próximo de 15 e comportamento também bastante previsível, sendo parte de benchmarks padrão que tratam de solução de sistemas de equações.

Para a execução da aplicação para cálculo de π pelo método

de Monte Carlo foram executados experimentos que têm como principal característica a manutenção da carga total fixa, ou seja, manteve-se constante a multiplicação entre o número de tarefas e o número de pontos em cada tarefa. Assim, foram feitos experimentos utilizando: 1.000 tarefas e 20.000.000 pontos por tarefa; 2.000 tarefas e 10.000.000 pontos por tarefa; 4.000 tarefas e 5.000.000 pontos por tarefa.

Para a execução da aplicação para multiplicação de matrizes, uma matriz de dimensão (4.000 X 1.000) foi multiplicada por uma matriz de dimensão (1.000 X 4.000), sendo o trabalho distribuído em 100, 200 e 400 tarefas.

Também neste caso a carga total é mantida constante, pois a primeira matriz é dividida em um número de submatrizes igual ao número de tarefas a serem executadas.

B. Avaliação do impacto no desempenho

Nas Tabelas I e II são exibidos dados obtidos nos experimentos realizados para observação do impacto da ferramenta de monitoramento JoiNMon no tempo de execução destas aplicações.

Os dados obtidos na primeira etapa, feita sem a ferramenta de monitoramento, estão na coluna intitulada *JoiNMon Inativo* e os dados obtidos na segunda etapa, com a ferramenta JoiNMon ativada, na coluna *JoiNMon ativo*.

Para estas duas etapas é apresentado o valor médio, em milissegundos, do tempo de processamento da aplicação obtido em cada uma das configurações utilizadas nos experimentos. Na coluna intitulada *Sobrecarga* são apresentados os percentuais de acréscimo que a ferramenta de monitoramento acarreta no tempo de execução destas aplicações.

Verificamos, nos dados obtidos nos experimentos realizados com a aplicação para o cálculo de π e apresentados na Tabela I, que a sobrecarga observada na execução não foi, em qualquer dos casos, superior a 3,54% e, na maioria (55%) dos experimentos foi inferior a 2,25%. A mediana dos valores obtidos é 2,11% e a porcentagem média de sobrecarga para todos os casos foi de 2,03%.

Neste experimento, nota-se uma tendência de queda na sobrecarga quando se aumenta o número de tarefas, o que sugere que a ferramenta de monitoramento não interfere de modo significativo no desempenho do sistema JoiN para a execução de aplicações com um grande número de tarefas e com $R_{cc} \geq 100$, que são as aplicações que melhor utilizam os recursos do sistema paralelo. A Figura 4(a) apresenta um gráfico onde pode ser observado que a sobrecarga imposta pela ferramenta de monitoramento não é significativa para a execução deste tipo de aplicação no sistema JoiN. Nota-se ainda a diminuição desta sobrecarga à medida que mais componentes *worker* são adicionados ao sistema. Com base nestes experimentos, concluímos que JoiNMon não interfere de modo significativo no desempenho do sistema JoiN para a execução de aplicações da classe $R_{cc} \geq 100$.

Verificamos, nos dados obtidos nos experimentos realizados com a aplicação para multiplicação de matrizes e apresentados na Tabela II, que a sobrecarga observada na execução não foi, em qualquer dos casos, superior a 5,52% e, na maioria (72%) dos experimentos foi inferior a 3,81%. A mediana dos valores obtidos é 3,47% e a porcentagem média de sobrecarga para todos os casos foi de 3,54%.

Também nestes experimentos há tendência de queda na sobrecarga de processamento quando se aumenta o número de tarefas. Isto pode ser verificado na maior parte das configurações utilizadas, o que sugere que a ferramenta de monitoramento também não interfere no desempenho do sistema JoiN para a execução desta classe de aplicações onde $1 < R_{cc} < 100$.

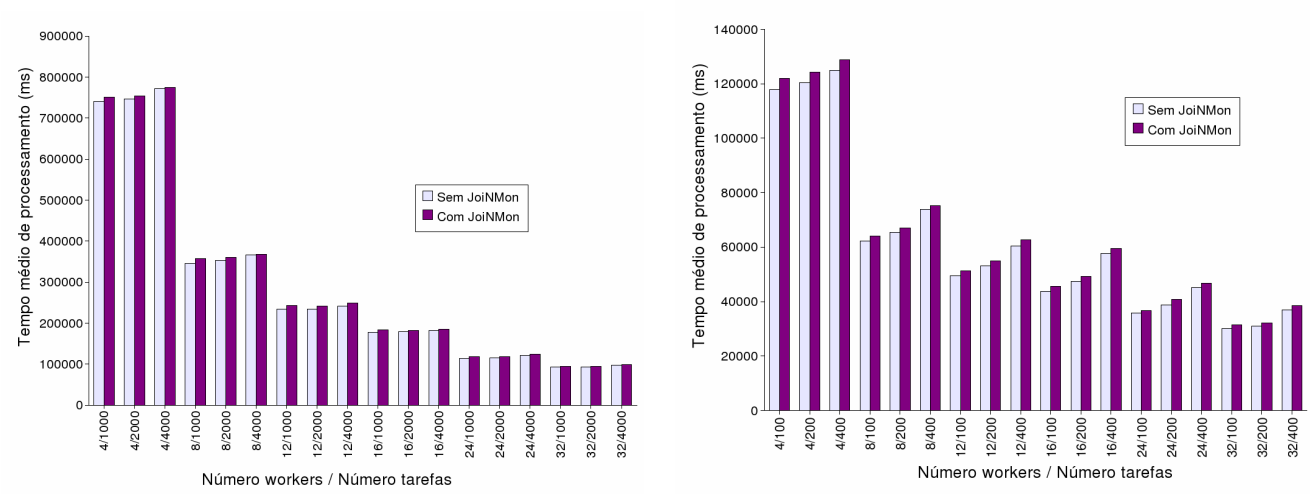
A Figura 4(b) apresenta um gráfico onde pode ser observado que a sobrecarga imposta pela ferramenta de monitoramento não é significativa em relação ao tempo total de processamento desta aplicação para multiplicação de matrizes.

Tabela I
CÁLCULO DE π - TEMPO MÉDIO PROCESSAMENTO (MS)

Número *workers*	Número tarefas	JoiNMon Inativo	JoiNMon Ativo	Sobrecarga
4	1000	740744	751638	1.47%
	2000	746631	754101	1.00%
	4000	771479	775437	0.51%
8	1000	345839	356818	3.17%
	2000	353007	360904	2.24%
	4000	366108	367573	0.40%
12	1000	234378	242675	3.54%
	2000	234806	242400	3.23%
	4000	242054	249095	2.91%
16	1000	177367	183127	3.25%
	2000	178814	182350	1.98%
	4000	183012	185463	1.34%
24	1000	114222	117796	3.13%
	2000	115706	119159	2.98%
	4000	121673	125140	2.85%
32	1000	93388	94679	1.38%
	2000	93837	94600	0.81%
	4000	98154	98563	0.42%

Tabela II
MULTIPLICAÇÃO DE MATRIZES - TEMPO MÉDIO PROCESSAMENTO (MS)

Número *workers*	Número tarefas	JoiNMon Inativo	JoiNMon Ativo	Sobrecarga
4	100	117918	121902	3,38%
	200	120471	124219	3,11%
	400	125064	128819	3,00%
8	100	62373	64013	2,63%
	200	65426	67069	2,51%
	400	73941	75260	1,78%
12	100	49414	51422	4,06%
	200	53150	55032	3,54%
	400	60408	62685	3,77%
16	100	43707	45589	4,31%
	200	47453	49255	3,80%
	400	57591	59549	3,40%
24	100	35776	36816	2,91%
	200	38708	40847	5,52%
	400	45068	46713	3,65%
32	100	30078	31523	4,81%
	200	31130	32142	3,25%
	400	36977	38600	4,39%



(a) Cálculo de π

(b) Multiplicação de matrizes

Figura 4. Tempo médio de processamento (ms)

Podemos constatar que, para a faixa de valores usados nestes experimentos, a ferramenta JoiNMon não introduziu gargalos no sistema e suportou o aumento do número de recursos (componentes *worker*) e de eventos (tarefas).

Em outras palavras, a implementação da ferramenta JoiNMon não aumentou significativamente o consumo de recursos disponíveis e, com isto, as aplicações não sofreram alterações perceptíveis no seu desempenho em função do monitoramento.

C. Avaliação do impacto no tráfego de rede

Durante a execução dos experimentos foram capturadas informações sobre o tráfego nas portas utilizadas pelo sistema JoiN e pela ferramenta de monitoramento. Esta captura foi feita por meio da ativação da ferramenta Wireshark no computador utilizado para a execução dos componentes *server* e *coordinator*.

A ferramenta Wireshark permite, além da captura das informações, a seleção e o resumo dos dados a serem analisados. Foram monitoradas as portas *default* utilizadas por JoiN e HSQLDB: TCP/8000, utilizada para comunicação entre os componentes *server* e *coordinator*; TCP/8100, utilizada para comunicação entre os componentes *coordinator* e *workers* a ele associados; TCP/9001, utilizada para acesso ao servidor de banco de dados (HSQLDB).

O tráfego de dados para transmissão de informações entre as tarefas que processam a aplicação para cálculo de π é muito pequeno, pois cada tarefa recebe um número como parâmetro (a quantidade de iterações que a tarefa deve realizar) e devolve um número como resultado (o número de iterações que satisfazem determinado critério). Para esta classe de aplicações ($R_{cc} \geq 100$) é esperado que a ferramenta de monitoramento introduza a maior sobrecarga relativa no tráfego de rede.

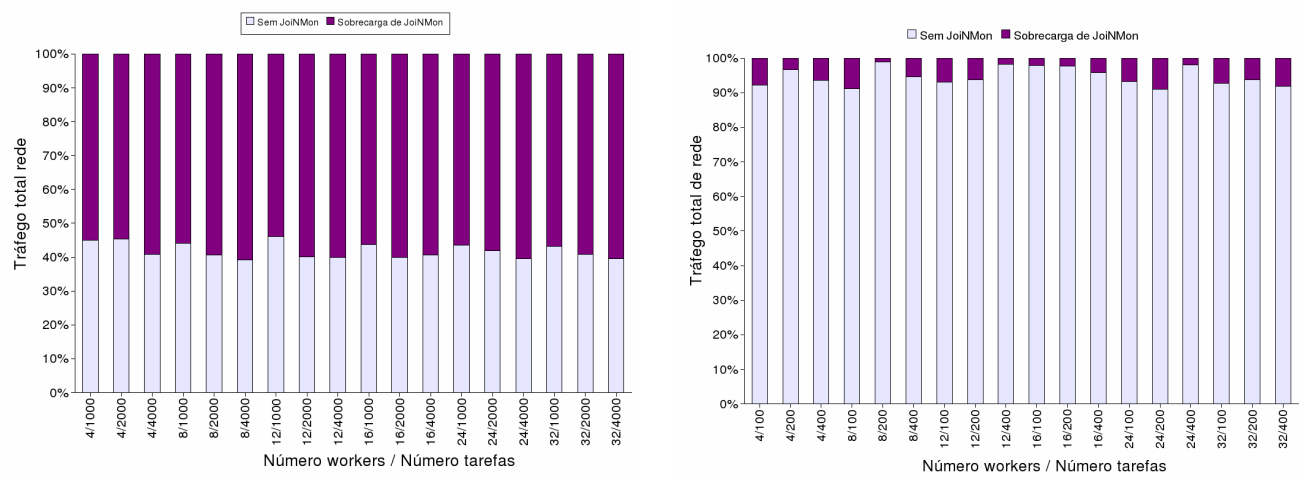
Nas Tabelas III e IV são exibidos os dados relativos ao tráfego de rede total das portas TCP/8000, TCP/8100 e TCP/9001 obtidos nos experimentos realizados. São apresentados os valores médios, em Kbytes, do tráfego de rede observado nestas portas durante a execução das aplicações e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante estas execuções.

Tabela III
CÁLCULO DE π - TRÁFEGO (KBYTES)

Número *workers*	Número tarefas	JoiNMon		Sobre-carga
		Inativo	Ativo	
4	1000	2245	4991	122,30%
	2000	4323	9542	120,69%
	4000	7587	18606	145,25%
8	1000	2174	4935	126,96%
	2000	3811	9375	146,00%
	4000	6932	17700	155,32%
12	1000	2228	4829	116,75%
	2000	3624	9083	149,36%
	4000	7039	17597	149,99%
16	1000	2114	4832	128,59%
	2000	3629	9079	150,19%
	4000	6866	16882	145,88%
24	1000	2026	4652	129,58%
	2000	3622	8628	138,21%
	4000	6766	17122	153,07%
32	1000	2089	4834	131,41%
	2000	3617	8875	145,38%
	4000	6672	16854	152,59%

A Figura 5(a) apresenta um gráfico onde pode ser observado que o tráfego de rede gerado pela ferramenta na execução desta aplicação está compreendido no intervalo de 55% a 60% do tráfego total observado durante a execução de uma aplicação da classe $R_{cc} \geq 100$ no sistema JoiN.

Os valores observados na aplicação para cálculo de π, parecem sugerir uma sobrecarga muito grande no tráfego de rede. Se considerarmos que, no contexto desta aplicação, o tráfego original é muito pequeno, podemos avaliar como satisfatórios os resultados obtidos, uma vez que, como já



(a) Cálculo de π

(b) Multiplicação de matrizes

Figura 5. Sobrecarga no tráfego de rede

Tabela IV
MULT. MATRIZES - TRÁFEGO (KBYTES)

Núm. *workers*	Número tarefas	JoiNMon		Sobre-carga
		Inativo	Ativo	
4	100	839315	911272	8.57%
	200	1372920	1420831	3.49%
	400	2064744	2208445	6.96%
8	100	874864	959550	9.68%
	200	1572440	1590308	1.14%
	400	2113837	2236642	5.81%
12	100	715696	769850	7.57%
	200	900301	959746	6.60%
	400	1415766	1443035	1.93%
16	100	713891	729732	2.22%
	200	830076	849674	2.36%
	400	1262175	1317042	4.35%
24	100	548677	588499	7.26%
	200	706469	776587	9.93%
	400	1127975	1150428	1.99%
32	100	667784	720471	7.89%
	200	786105	839407	6.78%
	400	1179037	1283914	8.90%

apresentado, a ferramenta de monitoramento não introduziu sobrecarga superior a 3,55% no tempo total de execução da aplicação, que inclui o tempo para comunicação.

Como R_{cc}= *tempo_de_computação / tempo_de_comunicação* e para esta aplicação $R_{cc} > 450$, se dobrássemos o tempo gasto em comunicação, ainda assim a execução da aplicação para cálculo de π no sistema JoiN apresentaria boa relação custo/benefício.

Entretanto, como as aplicações da classe $R_{cc} \geq 100$ não necessariamente apresentam um volume de tráfego de dados insignificante, é possível termos um impacto relativo reduzido da ferramenta de monitoramento no tráfego de rede se houver um certo volume de dados sendo transmitido, já que o tráfego extra de JoiNMon pode ser considerado estável para um mesmo número de tarefas.

O tráfego de dados para transmissão de informações entre as tarefas que processam a aplicação para multiplicação de matrizes pode ser grande pois para cada uma das tarefas são enviadas as matrizes a serem multiplicadas: são enviadas todas as linhas e colunas da primeira matriz; a segunda matriz é subdivida conforme o número de tarefas a serem executadas e cada porção da mesma é enviada para a tarefa correspondente. Cada tarefa retorna uma submatriz contendo o resultado da multiplicação. Para esta classe de aplicações ($1 < R_{cc} < 100$) é esperado que a sobrecarga relativa introduzida pela ferramenta de monitoramento no tráfego de rede não seja muito expressiva.

Na Tabela IV também podemos observar que a sobrecarga no tráfego de rede introduzida pela ferramenta de monitoramento durante os diversos experimentos conduzidos com a aplicacação para multiplicação de matrizes não chegou a 10%, o que representa um resultado satisfatório uma vez que, como já apresentado, a ferramenta de monitoramento não introduziu sobrecarga superior a 5,52% no tempo total de execução da aplicação, que inclui o tempo para comunicação. O valor médio da sobrecarga introduzida na comunicação foi de 5,75%.

A Figura 5(b) apresenta um gráfico onde pode ser observado que o tráfego de rede gerado pela ferramenta de monitoramento na execução desta aplicação representa no máximo 10% do tráfego total observado durante a execução de uma aplicação da classe $1 < R_{cc} < 100$ no sistema JoiN.

Analisando a Figura 6, onde é apresentado o gráfico correspondente ao tráfego médio por tarefa da aplicação para multiplicação de matrizes, podemos observar o mesmo comportamento em cada par de curvas, composto por: uma linha contínua, que representa o experimento sem a ferramenta de monitoramento; uma linha pontilhada, que representa o JoiNMon ativado. Pela observação das informações exibidas nesta figura, concluimos que a ferramenta de monitoramento

não altera significativamente o comportamento do tráfego de rede do sistema JoiN na execução desta aplicação.

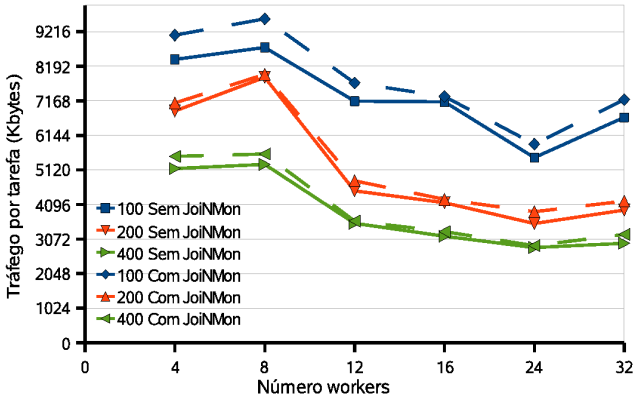


Figura 6. Multiplicação de matrizes - Tráfego médio por tarefa

V. Conclusões e trabalhos futuros

Este artigo apresentou uma proposta para monitoramento hierárquico de serviços e aplicações do sistema JoiN. Com a implementação desta proposta, os usuários e colaboradores passam a ter acesso a informações sobre o estado do sistema em tempo-real e também a informações sobre o histórico da execução de aplicações passadas.

Avaliações e testes preliminares, tanto de desempenho global como de impacto no tráfego de rede, com diferentes tipos de aplicações, mostraram que a ferramenta proposta praticamente não alterou o desempenho do sistema sob monitoramento e apresentou uma boa relacao custo/benefício. Para uma compreensão mais aprofundada do comportamento de JoinMon são necessários ainda testes mais abrangentes e detalhados. Mesmo assim, algumas melhorias já podem ser apontadas.

Dentre melhorias que podem ser sugeridas para aumentar a eficiência e a abrangência da ferramenta estão:

- inclusão de funcionalidades para monitoramento dos recursos de armazenamento e de rede utilizados pelo sistema JoiN;
- desenvolvimento de ferramentas para análise de desempenho e identificação de problemas;
- implementação de mecanismos para replicação das informações monitoradas para, desta forma, aumentar a tolerância a falhas;
- implementação de novas funcionalidades na interface gráfica relativas à apresentação de gráficos, de análises estatísticas e de informações gerenciais;
- implementação de mecanismos de tolerância a falhas no armazenamento dos dados coletados.

Apesar do artigo descrever a implementação bem sucedida de uma ferramenta de monitoramento para uma plataforma específica, entendemos que suas características, arquitetura e formas de avaliação podem servir de base para orientar o projeto e a implementação de ferramentas similares em outros tipos de sistemas paralelos ou distribuídos.

Agradecimentos

Pesquisa desenvolvida com o auxílio do CENAPAD-SP (Centro Nacional de Processamento de Alto Desempenho em São Paulo), projeto UNICAMP / MCTI.

Referências

[1] E. J. H. Yero. Estudo sobre processamento maciçamente paralelo na internet. Tese de doutorado, Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação, Julho 2003.

[2] E. J. H. Yero, F. de Oliveira Lucchese, F. S. Sambatti, M. von Zuben, and M. A. A. Henriques. Join: The implementation of a java-based massively parallel grid. *Future Generation Computer Systems*, 21(5):791–810, May 2005.

[3] D. P. Anderson. Public computing: Reconnecting people to science. *Conf. on Shared Knowledge and the Web*, November 2003. http://boinc.berkeley.edu/boinc2.pdf.

[4] M. L. Massie, B. N. Chun and D. E. Culler The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817-840, July 2004.

[5] H. B. Newman, I. C. Legrand, P. Galvez, R. Voicu and C. Cirstoiu MonALISA : A Distributed Monitoring Service Architecture *Proceedings of the Conference on Computing in High Energy and Nuclear Physics (CHEP 2003)*, March 2003. http://monalisa.caltech.edu/documentation/MOET001.pdf.

[6] LEMON - LHC Era Monitoring. Página na internet, Setembro 2013. http://lemon.web.cern.ch/lemon/index.shtml.

[7] B. Tierney, B. Crowley, D. Gunter, J. Lee and M. Thompson A Monitoring Sensor Management System for Grid Environments *Cluster Computing*, 4(1):19-28, 2001 http://www-didc.lbl.gov/papers/JAMM.HPDC00.pdf

[8] A. W. Cooke, A. J. G. Gray, L. Ma, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Byrom, L. Field, S. Hicks, J. Leake, M. Soni, A. J. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B. A. Coghlan, S. Kenny and D. O'Callaghan R-GMA: An Information Integration System for Grid Monitoring *Proceedings of the 11th International Conference on Cooperative Information Systems - CoopIS/DOA/ODBASE*, 2888:462-481, November 2003

[9] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski. A grid monitoring architecture. Informational document, GGF - Global Grid Forum, 2002. http://www.ggf.org/documents/GFD/GFD-I.7.pdf.

[10] Hsqldb database engine. Página na internet, Setembro 2013. http://www.hsqldb.org/.

[11] Wireshark. Página na internet, Setembro 2013. http://www.wireshark.org/.

[12] NAS Parallel Benchmarks. Página na internet, Setembro 2013. http://www.nas.nasa.gov/publications/npb.html/.