Avaliação de desempenho do sistema COSY: um sistema operacional concorrente para Transputers

César A. F. De Rose*
Philippe O. A. Navaux†

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Pós-Graduação em Ciência da Computação
Caixa Postal 15064
CEP 91501-970 - Porto Alegre, RS, Brasil
Tel.: (051)336-8399 Fax: (051)336-5576

Resumo

Com o aumento gradativo do número de elementos processadores nas máquinas multiprocessadoras atuais e a conseqüente necessidade de um gerenciamento eficaz destes recursos no sistema, cresce a demanda por sistemas operacionais multiprogramados que sejam capazes de lidar com este tipo de máquina. O sistema COSY foi desenvolvido na Universidade de Karlsruhe como uma resposta a esta demanda. Neste trabalho, é feita uma comparação entre o sistema COSY e os outros sistemas encontrados no mercado. Com base numa implementação do algoritmo paralelo de ordenação *Merge-Split-Sort*, é feita uma comparação direta entre os resultados obtidos pela primeira versão do sistema COSY e a linguagem OCCAM e são apresentados pontos a serem trabalhados para a melhoria do sistema.

Palavras-chave: Sistemas Operacionais Concorrentes, Processamento Paralelo, Máquinas Multiprocessadoras baseadas em Transputers

Abstract

The increase on the number of processor elements in todays multiprocessors and the resulting necessity of a eficient management of those resources in the system, raised the demand for multiprogramed operating systems that can handle this kind of machines. The COSY system was developed in the University of Karlsruhe as an response to this demand. This work makes a comparission among COSY and the other systems in the market. Based on a implementation of the parallel Merge-Split-Sort algorithm a direct comparission between the results obtained by a first COSY version and OCCAM are made and suggestions to improve the system are presented.

Keywords: Concurrent Operating Systems, Parallel Processing, Transputer based Multiprocessors

*Mestrado em Informática (CPGCC/UFRGS, 1993); Doutorando na Universidade Fridericiana de Karlsruhe; Áreas de Interesse: Arquitetura de Computadores, Sistemas Operacionais, Processamento Paralelo e Distribuído; E-mail: derose@ira.uka.de

†Professor UFRGS/CPGCC; Dr. Eng. em Informática (Instituto Nacional Politécnico de Grenoble, França,1979); Áreas de Interesse: Arquitetura de Computadores, Processamento Paralelo, Avaliação de Desempenho; E-mail: navaux@inf.ufrgs.br

1 Introdução

Nos últimos anos, máquinas paralelas (não somente baseadas em Transputers [INM89]) têm se tornado cada vez mais poderosas. Ao lado da utilização de processadores cada vez mais poderosos, também tem crescido o número de processadores utilizados nestes sistemas. Máquinas paralelas com mais de mil processadores já são uma realidade e máquinas com um número maior de processadores ainda são esperados em um futuro bem próximo.

Porém, o desenvolvimento de software não acompanhou o desenvolvimento do hardware. Para estas máquinas macissamente paralelas, praticamente não existe sistema operacional. Na maioria dos casos, a máquina é dividida em partições fixas, em cada uma das quais um sistema de tempo real ligado à aplicação é disparado. A multiprogramação dinâmica, que há décadas já é uma realidade em sistemas monoprocessados, permitindo um melhor aproveitamento dos recursos do sistema, é desta forma raramente utilizada.

Com o objetivo de preencher o vazio existente nesta área foi desenvolvido o sistema COSY (Concurrent Operating SYstem) [BUT92] [BUT94a] [BUT94b]. O projeto da Universidade de Karlsruhe tem por objetivo aplicar as técnicas mais modernas de gerência de processadores em sistemas dinâmicos, como as descritas em [DER93], na construção de um sistema operacional concorrente para máquinas macissamente paralelas. Com a conclusão da primeira versão deste sistema surge mais uma opção de ambiente de desenvolvimento para máquinas paralelas baseadas em Transputer.

Este trabalho pretende comparar o sistema COSY com os outros sistemas encontrados no mercado e analisar as vantagens e desvantagens de sua utilização em máquinas multiprocessadoras macissamente paralelas.

Com o objetivo de avaliar o desempenho da primeira versão do sistema COSY é analisado o desempenho obtido por uma aplicação paralela sob este sistema, no caso o algoritmo de ordenação *Merge-Split-Sort*. Os resultados obtidos são comparados com o desempenho do mesmo algoritmo implementado no ambiente de desenvolvimento OCCAM [INM90]. Ambas as versões paralelas foram desenvolvidas para uma máquina *Super-Cluster* da firma Parsytec com 64 Transputers. Com base na análise dos resultados obtidos são apresentadas sugestões de melhorias para futuras versões do sistema COSY.

Na seção 2 é apresentado o sistema operacional COSY e suas principais caracteríStcas. Uma avaliação dos sistemas encontrados no mercado para este tipo de máquina e a inserção do sistema COSY neste contexto é feita na seção 3. O algoritmo de ordenação *Merge-Split-Sort*, utilizado na avaliação do desempenho da primeira versão do sistema COSY é apresentado na seção 4, juntamente com alguns detalhes de sua implementação comuns aos dois ambientes de desenvolvimento utilizados. Os resultados obtidos são apresentados na seção 5 juntamente com sua análise e as características da máquina paralela baseada em Transputers utilizada. As conclusões deste trabalho são apresentadas na seção 6.

2 O Sistema Operacional COSY

Nesta seção é apresentada de forma superficial a estrutura do sistema COSY e suas principais características, dando uma atenção especial para a funcionalidade do núcleo do sistema e a camada de transporte. Informações detalhadas sobre o sistema e seus componentes podem ser encontradas em [BUT92] [BUT94a] [BUT94b].

2.1 Características gerais do sistema

A estrutura do sistema COSY pode ser vista na figura 1. Ela segue o modelo mundialmente aceito do núcleo com funções reduzidas, sobre o qual as funções extras são implementadas como processos ([GIE89] [MUL90]). Cada nodo do sistema possui um destes núcleos, mas nem todos os servidores se encontram em todos os nodos. Desta forma existe um gerenciador de memória presente em todos os nodos para a gerência da memória local, mas outros serviços, como a comunicação com a máquina hospedeira ou a gerência de arquivos, se encontram apenas nos nodos responsáveis por estas funções (na figura 1, os nodos 0 e 2 respectivamente).

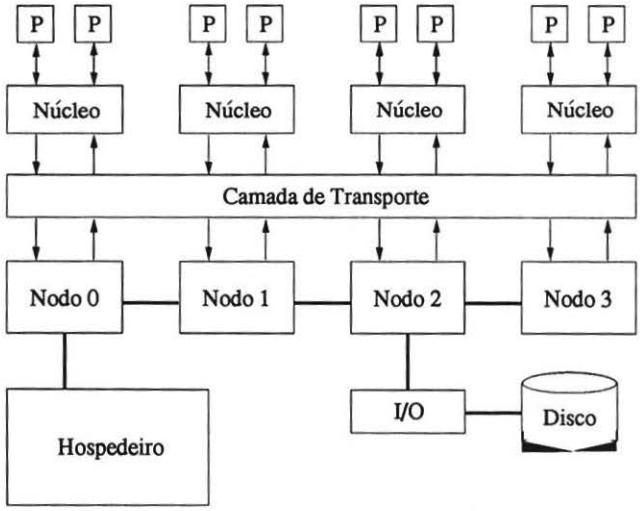

Figure 1: Estrutura do sistema COSY

Do ponto de vista do núcleo do sistema, não existe diferença entre servidores e processos de usuários. Isto permite, entre outras possibilidades, a ativação de servidores adicionais, ou a desativação de servidores existentes. A estrutura descentralizada resulta no atendimento local de requisições, uma das razões para a fácil escalabilidade do sistema.

A instância de transporte (roteador) não está integrada no núcleo, mas se localiza entre o núcleo e o hardware. Os principais motivos desta separação são a simplificação do núcleo e a independência do hardware. Máquinas paralelas baseadas em Transputers pertencem a uma pequena classe de máquinas nas quais o transporte de mensagens entre nodos intermediários tem que ser feito com ajuda do software do sistema. Porém, as máquinas mais novas já possuem hardware específico para esta tarefa, o que acelera consideravelmente esta função. Do ponto de vista do núcleo, a instância de transporte já é ativada através de uma interface reduzida, que pode ser facilmente adaptada na nova geração de máquinas com hardware específico para esta função.

2.2 Funcionalidade do núcleo

O núcleo tem a tarefa de viabilizar o acesso à infra-estrutura que permite a execução e interação de processos. O Transputer já tem um núcleo mínimo embutido, que é muito eficiente, mas serve apenas para um único programa. A ele faltam uma gerência de processos, *time-slices* variáveis, mais de duas prioridades e operações de comunicação mais seguras e mais confiáveis. Todas estas características estão disponíveis no núcleo do sistema COSY.

Para obter uma maior independência do hardware, o núcleo foi implementado na linguagem "C", somente na camada de entrada do núcleo e na comunicação com a camada de transporte foi utilizada a linguagem assembler. Para permitir um maior conforto na programação, todas as chamadas ao núcleo não são apenas locais, mas possíveis em qualquer parte do sistema. Desta forma, podem ser criados processos em nodos remotos e atributos destes processos podem ser lidos e setados. Logo depois da entrada no núcleo, é testado se a rotina desejada está disponível localmente. Se não for o caso, uma requisição é enviada para o local onde se encontra a rotina, a operação é executada de forma remota e os resultados são retornados. Tudo isto é feito de forma transparente para a aplicação que fez a chamada. Para este tipo de chamada, o núcleo se utiliza da camada de transporte que se encontra abaixo dele.

Como foi citado acima, COSY suporta várias prioridades de processos. Processos da mesma prioridade são executados se utilizando a técnica de fatias de tempo, com fatias de tamanho variável. As prioridades não são apenas setadas pela aplicação, mas podem ser alteradas pelo sistema de forma dinâmica. Desta forma, a prioridade de uma aplicação pode ser diminuída no final de sua fatia de tempo e aumentada, por exemplo, quando é desbloqueada (após o recebimento de uma mensagem que foi esperada de forma síncrona). Este mecanismo simples e eficiente resulta em um tempo de execução muito bom, mantendo alto o desempenho do sistema [BUT92].

O núcleo também oferece suporte para diversos tipos de medições. Um relógio com resolução de 1/100 segundos é mantido constantemente pelo sistema, permitindo a medição do tempo de execução real e a medição de tempo entre eventos. Além disto, é medido para cada processo o tempo de utilização do processador. Através de cáculos simples com estes valores, é possível avaliar o desempenho de diferentes algoritmos. Se, por exemplo, um processo executa sozinho em um nodo, é possível através da diferença destes valores, determinar quanto tempo este processo perdeu em comunicação e/ou sincronização. A diferença entre o tempo real e o tempo de CPU do processo *idle* permite uma análise rápida da carga de um nodo.

2.3 A camada de transporte

A transferência de pacotes entre nodos do sistema é feita pela camada de transporte. Para possibilitar a utilização transparente dos serviços do núcleo local para toda a rede de forma eficiente, a camada de transporte não pode se tornar o gargalo do sistema.

Em alguns sistemas híbridos, como por exemplo a máquina Parsytec GC, a camada de transporte é implementada por um grupo de processadores dedicados que, juntamente com um software de roteamento, se encarregam da comunicação entre nodos. O processador principal é liberado desta tarefa, e se encarrega exclusivamente das funções do sistema operacional e das aplicações.

Em um sistema composto apenas por nodos Transputer esta separação não é possível. Sendo assim, o processador divide seu tempo entre aplicações, funções do sistema operacional

e ainda tarefas da camada de transporte. Com o objetivo de manter o sistema estruturado em camadas e facilitar sua portabilidade, a camada de transporte foi implementada abaixo do núcleo, com a funcionalidade de um processador de comunicações. No caso do sistema ser portado para uma máquina com hardware dedicado para o transporte, esta camada é simplesmente descartada e o núcleo se comunica com o hardware dedicado diretamente através de sua interface.

Outro fator importante no desempenho da camada de transporte é a escolha do algoritmo de roteamento. Para este fim foi escolhido um algoritmo adaptativo de escolha de caminhos não mínimos, que não depende da topologia da máquina alvo. Decisões em relação ao caminho que um pacote deve seguir são tomadas de forma local à cada nodo, dependendo da carga atual das opções (adaptatividade). A existência de múltiplos caminhos mínimos entre o nodo origem e o nodo destino é desta forma aproveitada. Outra vantagem desta técnica é que regiões sobrecarregadas da rede de comunicação são evitadas através da utilização de desvios.

3 Comparando COSY com outros sistemas

Nesta seção é feita uma breve análise dos sistemas para máquinas paralelas disponíveis no mercado, juntamente com comparações a nível de funcionalidade e desempenho com o sistema COSY.

3.1 A situação atual do mercado

Para Transputers existe já há algum tempo o sistema HELIOS [PER91], que também foi portado para outras plataformas. Ele é baseado fortemente no sistema operacional Unix, mas não conseguiu vingar. Um dos motivos certamente foi a falta de escalabilidade do sistema, que dificulta principalmente o envio de mensagens em máquinas com um grande número de processadores. Como COSY, HELIOS é baseado no modelo de vários servidores mas não dispoẽ de um núcleo próprio, dependendo do núcleo simplificado dos Transputers. Algumas de suas deficiências podem ser atribuídas a este enfoque: processos só podem ter duas prioridades e somente processos de prioridade baixa são gerenciados com fatias de tempo fixas. Estas limitações resultam em tempos de resposta muito ruims, especialmente em máquinas muito carregadas.

Para máquinas macissamente paralelas, com um número de processadores entre 100 e 1000, foi desenvolvido pela firma Parsytec o ambiente de programação PARIX. Aqui não se trata de um sistema operacional propriamente dito, pois cada usuário recebe de um gerenciador de rede um grupo de processadores alocados, sob os quais uma aplicação é disparada juntamente com o sistema de tempo de execução PARIX. Para entrada e saída de dados é necessária a participação de uma máquina hospedeira. A divisão da máquina em partições fixas, onde rodam sistemas monousuários/monoprogramados otimizados e eficientes, resulta no ótimo desempenho para as operações de comunicação como é mostrado em [HEI92]. Por outro lado, a pouca flexibilidade na gerência dos processadores (não é permitida, por exemplo, a alocação de mais processadores em tempo de execução) e a forma que o "sistema operacional" é lincado à aplicação como uma biblioteca, lembram a gerência de sistemas dos anos 50. Estes fatores resultam em uma má utilização da máquina paralela e desperdício de poder computacional.

Como as aplicações para máquinas paralelas devem ser quebradas em no mínimo tantos processos quanto existem processadores disponíveis na máquina, a fim de explorar o maior grau de paralelismo possível, a comunicação entre processos tem um papel muito importante nestas máquinas. No mercado se encontram diferentes tentativas de se padronizar um modelo de comunicação. Uma destas tentativas é MPI (*message passing interface*) [MPI94], que em muitos pontos se assemelha com PVM (*parallel virtual machine*) [GEI93]. PVM foi originalmente desenvolvido para para redes de estações de trabalho, mas já foi portado também para máquinas paralelas.

Tanto MPI como PVM não se utilizam de canais como objetos de comunicação, mas endereçam mensagens diretamente a processos, com cada processo possuindo um número de identificação. A mensagem enviada recebe, além do número de seu destino, um outro identificador que pode ser livremente escolhido pelo emissor. Em uma operação de recebimento, a operação só é efetivada com uma mensagem de um determinado processo que possui uma determinada identificação. Ainda é possível receber uma mensagem de um processo qualquer com uma determinada identificação ou vice-versa. COSY por outro lado se utiliza de outro método de endereçamento. A utilização de canais como objetos de comunicação inclui mais um nível entre emissor e receptor. Esta abstração permite identificar serviços apenas pelo seu canal de recebimento de pedidos, por trás do qual podem ser encontrar diversos processos paralelos.

Na figura 2 é apresentada uma visão geral do mercado de sistema operacionais e ferramentas de desenvolvimento para máquinas macissamente paralelas em relação a sua funcionalidade e ao tamanho da máquina que são capazes de gerenciar.

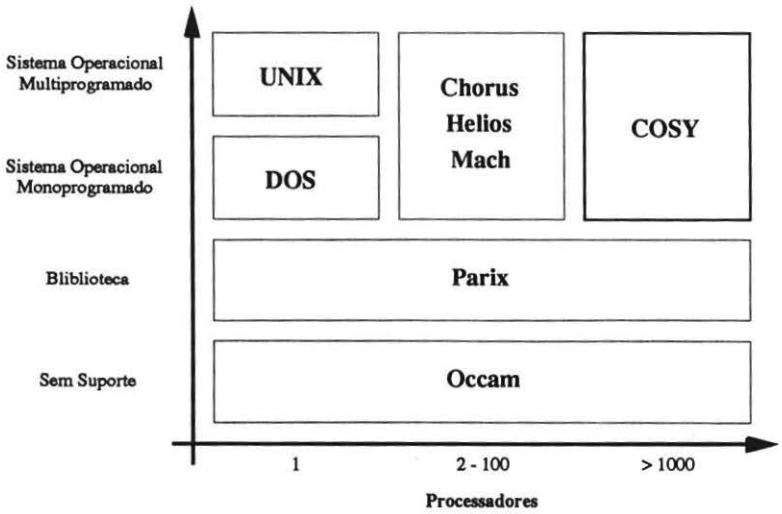

Figure 2: Os sistemas encontrados no mercado

3.2 Comparando o desempenho dos sistemas

Durante o desenvolvimento de uma aplicação paralela já se pode observar que a utilização de um sistema operacional traz vantagens em relação a um ambiente de programação como

PARIX ou OCCAM. Um exemplo é o disparo simultâneo de um monitor juntamente com a aplicação. Desta forma é possível observar a carga de um nodo do sistema ou da rede de comunicação. A multiprogramação com partição espacial e temporal da máquina paralela também resulta em um melhor aproveitamento dos recursos que a monoprogramação ou a partição estática da máquina.

A próxima pergunta é naturalmente se esta multiprogramação não tem um preço muito alto, ou seja, se o custo desta gerência não compromete o desempenho global do sistema. Para esclarecer esta questão são apresentadas na tabela 1 algumas medidas de desempenho comparando COSY com outras máquinas paralelas/sistemas operacionais. Juntamento com dados sobre comunicação e o tempo de uma operação ponto flutuante (IEEE, precisão 64 bits), é apresentada uma relação entre comunicação e cálculo. Os resultados demostram que COSY oferece, em relação ao tempo de cálculo, um ótimo desempenho de comunicação. É esperado que esta relação seja mantida com a utilização de processadores mais rápidos, pois grande parte do custo de comunicação resulta da execução de software no núcleo, que também será proporcionalmente acelerada.

Table 1: Compração de desempenho com outros sistemas (tempo em μ-segundos)

Sistema	latência no envio de uma mensagem	tempo por byte transferido	multiplicação ponto flutuante	relação comunicação/cálculo
iPSC/860	470	0.77	0.025	30.80
Intel Paragon	60	0.05	0.015	3.33
nCUBE2	210	0.58	0.5	1.16
COSY no T805	195	0.59	0.667	0.88

4 O algoritmo de ordenação *Merge-Split-Sort*

Nesta seção é descrito o funcionamento do algoritmo de ordenação *Merge-Split-Sort* e ressaltadas suas principais características. Também é apresentado o modelo utilizado na implementação do algoritmo, a forma com que os processos foram alocados na máquina paralela e detalhes de implementação comuns a ambas as plataformas de desenvolvimento.

Características da máquina paralela utilizada e uma descrição dos diferentes ambientes de programação, bem como as diferenças entre as implementações em COSY e OCCAM, são apresentados na seção 5.

4.1 Funcionamento do algoritmo e suas principais características

O algoritmo paralelo de ordenação *Merge-Split-Sort* pode ser visto como uma adaptação do algoritmo *Odd-Even-Transposition-Sort* que vem a permitir a variação da carga de trabalho dos nodos envolvidos no procedimento de ordenação [AKL85] [MOL93]. Desta forma é possível variar a granulosidade da operação elementar de troca de dados entre vizinhos, possibilitando uma melhor adaptação do algoritmo à máquina alvo.

No algoritmo original, que foi desenvolvido para máquinas SIMD [FLY72], era necessário o mesmo número de elementos processadores (EP) quanto for o número de elementos do vetor a ser ordenado. Cada EP armazena um elemento do vetor em sua memória local e a operação de troca com os vizinhos se resume a apenas um elemento.

O algoritmo *Merge-Split-Sort* parte do princípio de que cada EP possui memória local suficiente para armazenar uma parte maior do vetor a ser ordenado e que também pode executar tarefas mais complexas do que lidar apenas com um elemento do vetor. Sendo p o número de EP e n o número de elementos do vetor a ser ordenado, cada EP $P_1, P_2, \ldots, P_n$ trabalha com uma parte do vetor (sub-vetor) de tamanho $\frac{n}{p}$ (nos exemplos é assumido que n é múltiplo de p, caso contrário o vetor é preenchido com zeros no seu início). O algoritmo funciona da seguinte forma:

1. Antes de se iniciar com os passos do algoritmo propriamente ditos, é necessário que cada EP ordene sequencialmente o seu sub-vetor;
2. Em um passo ímpar, todos os EP $P_i (1 \leq i < p)$ com índice i ímpar estão ativos. Eles requisitam o sub-vetor do vizinho $P_i + 1$, juntam com o sub-vetor local e reordenam este novo sub-vetor de tamanho $2.\frac{n}{p}$ (fase *Merge*). Este novo sub-vetor é então separado novamente, com os primeiros $\frac{n}{p}$ ficando no EP P_i e o restante sendo enviado para $P_i + 1$ (fase *Split*);
3. Em um passo par, todos os EP com índice par estão ativos e efetuam um *Merge* e um *Split* com o seu vizinho da direita;
4. Os passos 2 e 3 são repetidos p vezes de forma intercalada.

No final da execução dos p passos os elementos estão ordenados de forma global. Isto significa que cada EP tem o seu sub-vetor ordenado e que os elementos de P_i são menores que os de $P_i + 1$ para $1 \leq i < p$.

A execução do algoritmo em um exemplo com vetor de tamanho 12 ($n = 12$) e quatro EP ($p = 4$) pode ser acompanhada na figura 3.

As principais características deste algoritmo são sua simplicidade, eficiência, e escalabilidade tanto em relação ao número de EP utilizados, quanto ao tamanho do vetor a ser ordenado. Outro fator muito importante é a possibilidade de se alterar a carga de processamento em uma EP e a carga de comunicação entre EP através da alteração de n e p.

Estas características foram decisivas na escolha do algoritmo de ordenação *Merge-Split-Sort* para a comparação de resultados entre sistemas. O algoritmo é facil de ser implementado e permite, com a simples alteração dos valores de n e p, a análise do desempenho dos sistemas em questão sob diferentes condições de número de nodos e sua carga e tamanho das mensagens.

4.2 Implementação nos Transputers

A implementação do algoritmo *Maerge-Split-Sort* foi realizada com o modelo mestre-escravo. O processo mestre é responsável pela divisão do vetor a ser ordenado, que se encontra na máquina hospedeira, em p sub-vetores de tamanho $\frac{n}{p}$ e pela distribuição destes sub-vetores para os respectivos p processos escravos. Os processos escravos executam os passos do algoritmo como descrito no ítem acima e devolvem seus sub-vetores para o processo mestre, que por sua vez reconstrói o vetor original. Na fase de coleta dos resultados e recomposição do vetor original, é importante que os sub-vetores não percam a sua ordenação.

A figura 4 apresenta o modelo utilizado e a forma como os processos foram interligados.

Como os processos escravos só precisam se comunicar com seus vizinhos da direita e esquerda, a topologia escolhida foi um vetor de processadores. Cada EP P_i só está ligado

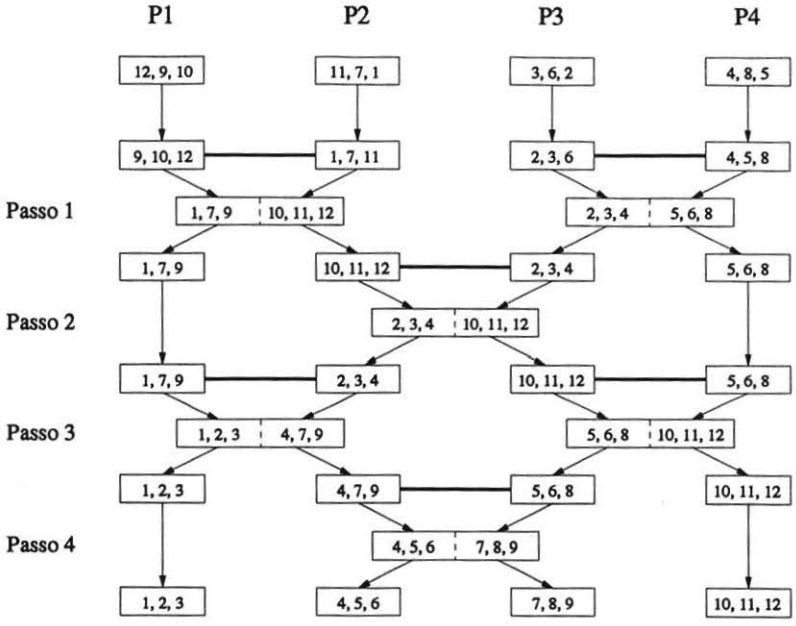

Figure 3: Algoritmo de ordenação *Merge-Split-Sort*

com o seu antecessor $P_i - 1$ e seu sucessor $P_i + 1$. Cada processo escravo foi alocado a um EP do vetor de processadores e o processo mestre foi alocado no primeiro EP do vetor, juntamente com um processo escravo. Este nodo é o único que possui uma ligação física com o hospedeiro.

Como o processo mestre se encontra no primeiro EP do vetor de processadores e os EP restantes só possuem ligações com seus vizinhos da esquerda e da direita, a carga e descarga deste vetor de processadores com dados é feita como em um *pipeline*. Na carga, o processo mestre alimenta o vetor de processadores pela esquerda e os dados são passados adiante até atingirem seu destino. A descarga dos dados é feita de forma análoga, apenas invertendo-se o sentido da operação (direita-esquerda).

A ordenação local dos elementos em um processo escravo é feita com o algoritmo *Bubble Sort* [KNU73]. Este algoritmo se baseia na troca entre elementos vizinhos do vetor a ser ordenado até que não existam mais pares fora de ordem. A cada passagem do algoritmo, elementos com valor mais alto são empurrados para o final do vetor e elementos com valor menor para seu início. Este algoritmo foi escolhido por ser de fácil implementação.

5 Avaliação da primeira versão do sistema COSY

Nesta seção são apresentados os resultados obtidos na avaliação da primeira versão do sistema COSY. Estes resultados foram obtidos através da comparação de duas versões paralelas do algoritmo *Merge-Split-Sort*, uma implementada para a primeira versão do sistema COSY e a outra em OCCAM. Também são apresentadas a máquina paralela utilizada e as características

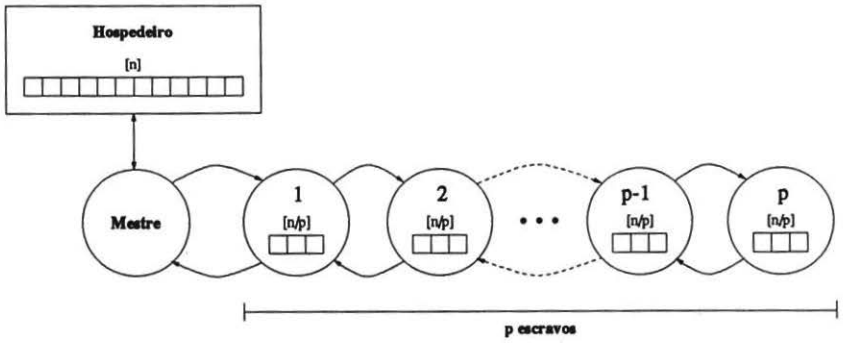

Figure 4: Implementação do algoritmo *Merge-Split-Sort*

particulares dos ambientes de programação.

5.1 A máquina paralela e os ambientes de programação

Para as medições de desempenho apresentadas neste trabalho foi utilizada uma máquina paralela baseada em Transputers do tipo *Super-Cluster* da firma Parsytec com 64 processadores [PAR89] (figura 5 (b)).

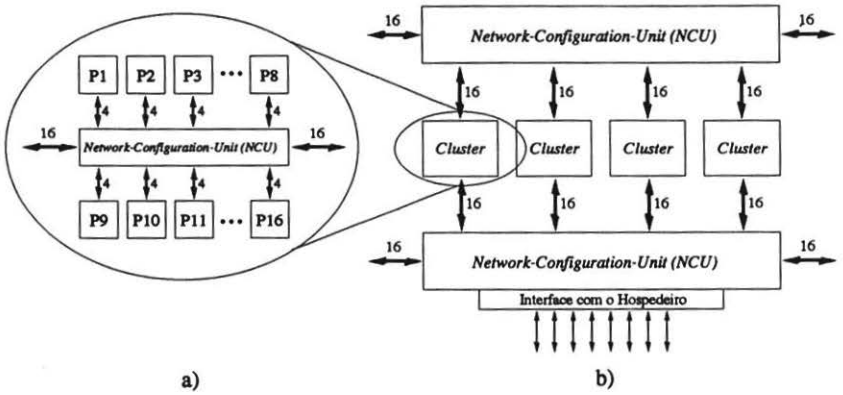

Figure 5: Máquina paralela baseada em Transputers utilizada nas implementações: a) Cluster, b) Super-Cluster

Os Transputers da máquina *Super-Clsuter* são organizados em dois níveis de forma hierárquica. No primeiro nível, 16 Transputers T800 com 4 Megabytes de memória local cada um, são ligados através de uma *Network Configuration Unit* - NCU compondo um *Cluster* (figura 5 (a)). A NCU pode ligar os 64 *links* dos 16 Transputers na topologia desejada e dispõe de 2 grupos de 16 *links* para a comunicação com outros *Clusters*. Desta forma, uma NCU realiza a função de uma chave *Crossbar* de tamanho 96x96 [HWA84]. No segundo nível, 4 destes *Clusters* são interconectados através de mais duas NCUs formando um *Super*

Cluster. Através desta arquitetura hierárquica se obtém uma maior flexibilidade na interconexão dos 64 Transputers. A conexão da máquina com o hospedeiro é feita através de 8 Transputers adicionais que estão integrados na estação de trabalho SUN que desempenha esta função. A estação tem como função a gerência de arquivos e comunicação com o usuário através de janelas.

Na máquina descrita acima foram instalados o sistema operacional COSY e o ambiente de programação *Occam-Toolset*. Os programas desenvolvidos para o sistema COSY são escritos em "C" convencional adicionado das chamadas ao sistema. Estes programas são compilados por um compilador "C" padrão e lincados com as bibliotecas do sistema. Os programas em OCCAM são compilados e lincados pelas ferramentas disponíveis no ambiente *Occam-Toolset*.

Ambas as versões do algoritmo *Merge-SPlit-Sort* se baseiam na descrição da seção 4 mas foram otimizadas ao máximo para os respectivos ambientes, o que resulta em pequenas diferenças entre as versões. A versão do algoritmo escrita em OCCAM, por exemplo, se aproveita da possibilidade de acesso ao hardware dos Transputers e otimiza ao máximo a carga e descarga dos dados com o uso paralelo dos canais de comunicação nestas operações. Outra diferença é que a versão para o sistema COSY precisa quebrar as mensagens do algortimo quando estas superam o limite de 4 kbytes. Esta limitação é imposta pelo algoritmo de roteamento da camada de transporte. Devido a maior funcionalidade do sistema COSY, algumas tarefas, feitas manualmente no sistema OCCAM, são feitas pelo processo mestre em tempo de execução, como a criação e estabelecimento das ligações entre os escravos (este tempo é naturalmente descontado nas comparações).

A linguagem OCCAM foi escolhida como parâmetro de comparação para esta primeira versão do sistema COSY por se tratar de uma linguagem que foi especialmente desenvolvida para trabalhar com Transputers. Sendo assim permite uma grande flexibilidade no acesso ao hardware e consequentemente um ótimo desempenho. A comparação do desempenho do sistema COSY com o melhor desempenho possível fornece uma boa idéia do custo resultante da maior funcionalidade que o sistema oferece (comparação funcional da seção 3).

5.2 Os resultados obtidos

A tabela 2 apresenta os resultados obtidos pelas duas implementações paralelas do algoritmo de ordenação *Merg-Split-Sort*. Os resultados correspondem ao tempo de execução do algoritmo na ordenação de um vetor de 12000 inteiros e a unidade de tempo é *ticks* do processador (um *tick* do processador corresponde a $64x10^{-6}$ segundos em baixa prioridade). Os resultados ainda estão divididos em tempo gasto com o cálculo (T_{calc}), tempo gasto com a comunicação ($T_{i/o}$) e tempo total de execução (T_{tot}), para permitir uma melhor análise dos resultados em ambos os sistemas.

A linha SUp_{rel} apresenta o *Speed-Up* relativo de cada execução que é obtido através da divisão do tempo de execução com 1 processador pelo tempo de execução com n processadores [HWA84]. A tempo de execução da versão OCCAM para um processador foi utilizado como referência (igual a 1) para permitir uma comparação a nível de *Speed-Up* entre a implementação nos dois ambientes.

O alto *Speed-Up* atingido pela paralelização do algoritmo e o seu comportamento superlinear resultam da utilização do algoritmo trivial de ordenação *Bubble-Sort* e da abordagem *worst-case* dos dados utilizados na execução do algoritmo (vetor a ser ordenado corresponde a ordenação decrescente dos elementos). Sendo assim o algoritmo paralelo de ordenação tem um desempenho muito melhor que o algoritmo seqüencial utilizado no cálculo do *Speed-Up*.

Table 2: Resultados obtidos na ordenação de um vetor de 12k inteiros (em *ticks*)

Processadores Escravos		1	2	4	8	16	32
COSY	T_{calc}	14064949	3516486	876897	423703	55509	15120
	$T_{i/o}$	2023	2141	3257	2400	11035	14633
	T_{tot}	14066972	3518627	880154	426103	66544	29753
	SUp_{rel}	0.55	2.18	8.73	18.03	115.47	258.25
OCCAM	T_{calc}	7861996	1920772	469767	118027	29695	6774
	$T_{i/o}$	1771	2266	2738	2761	2774	3422
	T_{tot}	7683767	1923038	472505	120788	32469	10196
	SUp_{rel}	1	4	16.26	63.61	236.65	753.61

O ganho obtido é ainda maior devido ao fato dos dados se encontrarem na pior situação possível para o algoritmo *Bubble-Sort*.

O gráfico da figura 6 mostra como se comportou o custo de comunicação com o aumento do número de processadores utilizados, para ambas as implementações. Como podemos ver, o custo de comunicação da versão para o sistema COSY se manteve similar ao da versão OCCAM até 8 processadores. Para as versões com mais de 8 processadores o custo dispara quase que de forma exponencial. Isto é resultado do custo adicional do algoritmo de roteamento da camada de transporte, que se torna mais significativo na medida que o número de mensagens do algoritmo aumenta e seu tamanho diminui. Outro fator a ser considerado é a necessidade da utilização do segundo nível de NCU's, quando o número de processadores envolvidos no algoritmo passa de 16. Isto resulta em um custo de comunicação extra para ambas as versões. Entretanto, o custo para a versão OCCAM é menor devido a sua proximidade com o hardware da máquina. Este é o ponto principal onde esta primeira versão do sistema tem que ser melhorada. Este comportamento exponencial do custo de comunicação é inaceitável para um sistema que se propõe a trabalhar com máquinas de até 1000 processadores.

O gráfico da figura 7 mostra como se comportou o custo total de execução de ambas as versões com o aumento do número de processadores utilizados. Aqui é possível observar uma clara vantagem para a versão implementada em OCCAM de um fator de 2 a 3 vezes. Esta diferença resulta da forma otimizada com que a linguagem OCCAM acessa a memória dos Transputers. Como o algoritmo de ordenação local se baseia fortemente na troca de valores entre endereços de memória, a diferença dos tempos de execução foi acentuada.

A perda de desempenho resultante do uso do sistema operacional COSY na implementação e execução do algoritmo de ordenação, em relação a implementação OCCAM, é a uma conseqüência direta do aumento de funcionalidade fornecido por este sistema. Considerando o exemplo apresentado e a necessidade de uma maior flexibilidade na gerência dos recursos de uma máquina paralela, o custo adicionado pelo sistema COSY é certamente um custo aceitável.

Ainda devemos considerar que o sistema continua em desenvolvimento e esta primeira versão será certamente melhorada, o que refletirá na qualidade dos resultados fornecidos por versões futuras do sistema.

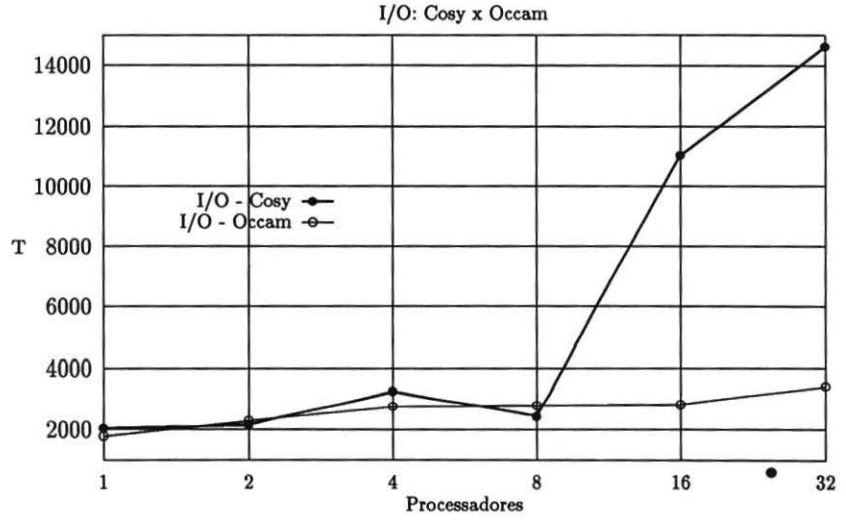

Figure 6: Desempenho da comunicação do algoritmo de ordenação

6 Conclusões

Neste artigo foi feita uma comparação do sistema operacional concorrente COSY, desenvolvido pela Universidade de Karlsruhe para máquinas macissamente paralelas baseadas em Transputers, com outros sistemas operacionais ou ambientes de programação encontradas no mercado para este tipo de máquina. Além da comparação funcional feita com outros sistemas, são apresentados resultados do desempenho obtido pela implementação do algoritmo de ordenação *Merge-Split-Sort* com a primeira versão do sistema COSY e com a linguagem OCCAM. Desta forma, é analisado o desempenho da primeira versão do sistema COSY e avaliado o custo resultante do aumento de funcionalidade trazido pelo uso deste sistema. Com base nos resultados obtidos são feitas sugestões de melhorias para as futuras versões deste sistema.

Por se tratar de um sistema operacional propriamente dito, estruturado em camadas e desenvolvido em base nas técnicas mais modernas de gerência de recursos e de comunicação entre processos, a utilização do sistema COSY resulta em um ganho de funcionalidade em relação as outras soluções encontradas no mercado. Dentro deste aumento de funcionalidade merecem destaque a (i) capacidade de multiprocessamento nos nodos do sistema, (ii) uma gerência de processos que permite a criação e alocação de processos em tempo de execução em qualquer nodo do sistema e (iii) uma camada de transporte que, se baseando na utilização de canais como objetos de comunicação e de um algoritmo eficiente de roteamento de pacotes, permite uma grande flexibilidade na interligação e interação entre processos de uma aplicação paralela.

É natural que um aumento de funcionalidade deste porte cause uma certa perda de desempenho nas aplicações implementadas e executadas sob o sistema operacional. A questão a ser analisada é se esta perda de desempenho, no caso do sistema COSY, pode ser considerada aceitável em uma máquina paralela. A comparação dos resultados de desempenho

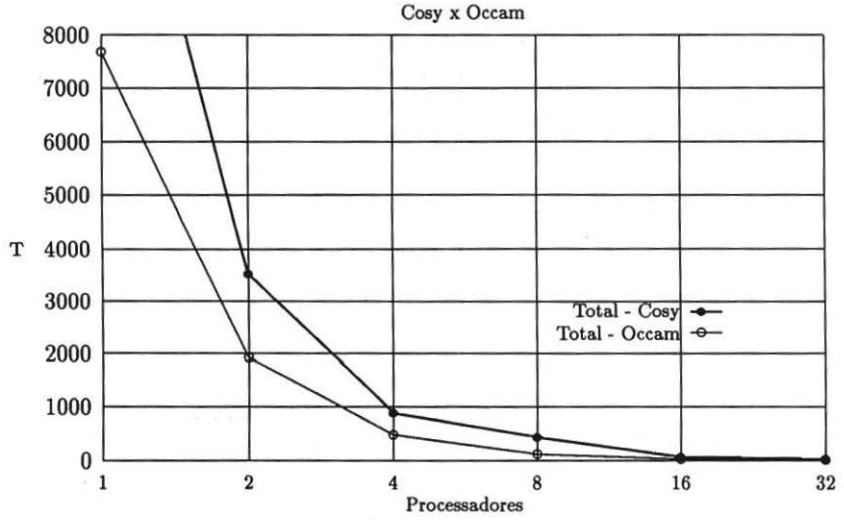

Figure 7: Desempenho do algoritmo de ordenação (unidade mil ticks)

obtidos com a utilização da primeira versão do sistema COSY com a aplicação em OCCAM mostrou uma perda de desempenho de um fator de 2 a 3 vezes se analisado o tempo total de execução da aplicação paralela. O custo de comunicação se manteve similar até 8 processadores, disparando de forma exponencial com o aumento do número de processadores acima desta marca.

Um fator importante a ser considerado é que o sistema COSY continua em desenvolvimento. Com base na primeira versão do sistema avaliada neste trabalho, estão sendo feitas diversas otimizações, principalmente na camada de transporte do sistema, com o objetivo de diminuir os custos com o roteamento de pacotes. Devido a portabilidade do sistema, está sendo considerada a implementação de uma versão para a nova geração de máquinas com o processador *Power-PC*.

O sistema operacional concorrente COSY vem com certeza contribuir para a área de software básico para máquinas multiprocessadoras, adicionando funcionalidade a este tipo de sistema e aumentando as possibilidades na gerência dos recursos destas máquinas. COSY é certamente o primeiro passo na direção de um sistema operacional robusto, capaz de gerenciar com eficiência e eficácia os recursos de máquinas multiprocessadoras macissamente paralelas.

References

[AKL85] AKL, S. G. **Parallel Sorting Algorithms**. Orlando, Florida: Academic Press. 1985.

[BUT92] BUTENUTH, Roger **KBS - An Operating System for a Small Computer**. Karlsruhe, Dep. of Informatics, University of Karlsruhe, 1992. (Diploma-Thesis)

[BUT94a] BUTENUTH, Roger The COSY-Kernel as an Example for Efficient Kernel Call Mechanisms on Transputers. **6th Transputer/Occam International Conference. Procedings...** Tokio. June, 1994.

[BUT94b] BUTENUTH, Roger; GILLES, Sven COSY - ein Betriebssystem für hochparallele Computer **TAT '94. Procedings...** Aachen. September, 1994.

[DER93] DE ROSE, César A. F. **Gerência e Alocação de Processadores em Máquinas Multiprocessadoras Hipercúbicas**. Porto Alegre, CPGCC da UFRGS, 1993. (Dissertação de Mestrado)

[FLY72] FLYNN, M. J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, New York, v. C-21, n. 9. pp. 948-160, Sept. 1972.

[GEI93] GEIST, A. et all **PVM 3.0 User's Guide and Reference Manual**. Technical Report ORNL/RM-12187. Oak Ridge National Laboratory. 1993.

[GIE89] GIEN, M. **Micro-Kernel Architecture: Key to Modern Operating System Design**. Chorus Systemes, technical report CS/TR-89-37.3.

[HEI92] HEISS, H.U. **Processor Management in Two-Domensional Grid-Architectures**. Internal Report 20/92. University of Karlsruhe. December 1992.

[HWA84] HWANG, K.; BRIGGS, F. **Computer Architecture and Parallel Processing**. New York: McGraw Hill. 1984.

[INM89] INMOS Limited. **The Transputer Databook (Second Edition)**. 1989.

[INM90] INMOS Limited. **Transputer Development System – Delivery Manual**. 1990.

[KNU73] KNUTH, Donald E. **The Art of Computer Programming – Sorting and Searching (Vol. 3)**. Menlo Park, California: Addison Wesley. 1973.

[MOL93] MOLDOVAN, Dan I. **Parallel Processing – From Applications to Systems**. San Mateo, CA: Morgan Kaufmann Publishers. 1993.

[MPI94] MPI Forum **MPI: A message-passing interface standart**. Technical Report. University of Tennesee. Mai 1994.

[MUL90] MULLENDER, et all AMoeba: A DIstributed Operating System for the 1990's. **IEEE Computer**, pp. 44-53, May. 1990.

[PAR89] Parsytec GmbH. **SuperCluster Technical Documentation, Revision 1.2**. 1989.

[PER91] PERIHELION **The Helios Parallel Operating System** Prentice Hall. 1991.