República Federativa do Brasil
Ministério do Desenvolvimento, Indústria e do Comércio Exterior
Instituto Nacional da Propriedade Industrial.

(21) **PI0613946-9 A2**

BRPI0613946A2

(22) Data de Depósito: 28/07/2006
(43) Data da Publicação: 22/02/2011
(RPI 2094)

(51) Int.Cl.:
G06F 13/00

(54) Título: **MEMÓRIA TRANSACIONAL DE SOFTWARE DE ATUALIZAÇÃO DIRETA**

(30) Prioridade Unionista: 29/07/2005 US 11/192.784

(73) Titular(es): MICROSOFT CORPORATION

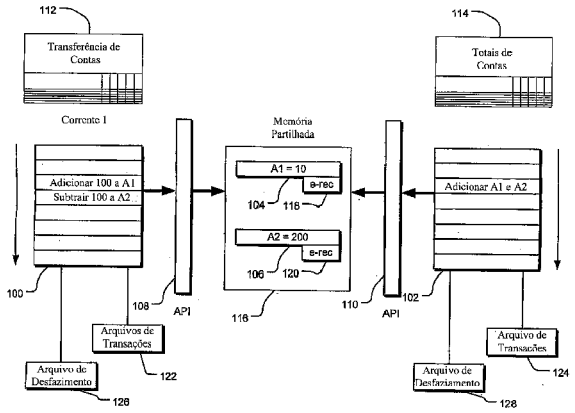
(72) Inventor(es): TIMOTHY L. HARRIS

(74) Procurador(es): Nellie Anne Daniel Shores

(86) Pedido Internacional: PCT US2006029327 de 28/07/2006

(87) Publicação Internacional: WO 2007/016302de 08/02/2007

(57) Resumo: MEMÓRIA TRANSACIONAL DE SOFTWARE DE ATUALIZAÇAO DIRETA. Uma interface de programação de memória transacional permite que uma corrente acesse, diretamente e com segurança, um ou mais locais de memória partilhada dentro de uma transação, enquanto mantendo estruturas de controle para gerenciar acessos de memória àqueles mesmos locais por uma ou mais correntes concorrentes. Cada local de memória acessado pela corrente é associado com um registro de listagem, e cada corrente mantém um arquivo de transações dos seus acessos de memória. Dentro de uma transação, uma operação de leitura é conduzida diretamente no local de memória, e uma operação de escrita é tentada diretamente no local de memória, em oposição a algum armazenamento temporário. A corrente pode detectar inconsistências entre o registro de listagem de um local de memória e o arquivo de transações da corrente, para determinar se os acessos de memória dentro da transação não são confiáveis e se a transação precisa ser tentada de novo.



PI0613946-9

"MEMÓRIA TRANSACIONAL DE SOFTWARE DE ATUALIZAÇÃO DIRETA".

CAMPO TÉCNICO

A invenção se refere, de uma maneira geral, a operações de memória de computador e, mais particularmente, à memória transacional de software de atualização direta.

ANTECEDENTES

É comum que cadeias múltiplas de processo multicadeia partilhem locais de memória comuns, durante execução concorrente. Conseqüentemente, duas diferentes cadeias de um programa multicadeia podem ler e atualizar os mesmos locais de memória, acessíveis pelo programa. No entanto, deve-se tomar cuidado para garantir que uma corrente não modifica um valor do local de memória partilhada, enquanto que a outra cadeia está na parte intermediária de uma seqüência de operações, que dependem do valor.

Por exemplo, supor que um programa está acessando o conteúdo de dois diferentes objetos de software, em que cada objeto representa uma quantidade de dinheiro em uma diferente conta bancária. Inicialmente, a quantidade da primeira conta é $10, armazenada no endereço de memória A1, enquanto que a quantidade da segunda conta é $200, armazenada no endereço de memória A2. Uma primeira corrente de um programa de transações bancárias é codificada para transferir $100 de A2 para A1, e uma segunda corrente é codificada para calcular a quantidade total de fundos em ambas as contas. A primeira corrente pode iniciar por adição de $100 ao conteúdo de A1, atualizando-o para $110, e depois continuar para

subtrair $100 do conteúdo de A2, atualizando-o para $100. No entanto, se a segunda corrente for executada entre essas duas operações, então a segunda corrente pode computar um total incorreto de $310 para ambas as contas, em vez do total correto de $210.

Uma memória transacional de software proporciona uma abstração de programação, através da qual uma corrente pode executar com segurança uma série de acessos de memória partilhada, permitindo que a corrente complete a sua transação sem interferência da outra corrente. Conseqüentemente, memórias transacionais podem ser empregadas em software para garantir que a transação, incluindo as operações exemplificativas de adição e subtração, da primeira corrente seja "atômica" quanto aos locais de memória A1 e A2, e, portanto, a segunda corrente vai computar a quantidade total correta em ambas as contas.

No entanto, as abordagens existentes para implementação de memórias transacionais em software sofrem de problemas de desempenho. Por exemplo, em uma abordagem existente, quando uma corrente acessa uma seqüência de locais de memória dentro de uma transação, a corrente mantém uma lista separada dos locais de memória e dos valores que deseja ler e atualizar (isto é, escrever), durante a transação, e depois, ao final da transação, a corrente atualiza todos esses valores nos locais de memória partilhada efetivos. Se, durante a transação, a corrente desejar reler ou reescrever em qualquer local de memória na sua lista, a corrente deve buscar a entrada do local da memória na lista para acessar a

entrada, que é, programaticamente, uma proposição lenta. Conseqüentemente, esse método indireto de implementação de uma memória transacional em software sofre de desempenho inferior.

SUMÁRIO

As implementações descritas e obtidas aqui abordam os problemas acima, proporcionando uma interface de programação de memória transacional, que permite que uma corrente acesse, diretamente e com segurança, um ou mais locais de memória partilhada dentro de uma transação, enquanto mantendo estruturas de controle para gerenciar os acessos de memória àqueles mesmos locais por uma ou mais outras correntes concorrentes. Cada local de memória acessado pela corrente é associado com um registro de listagem, e cada corrente mantém um arquivo de transações dos seus acessos de memória. Dentro de uma transação, uma operação de leitura é feita diretamente no local de memória, e uma operação de escrita é tentada diretamente no local de memória, diferentemente de em algum armazenamento temporário intermediário. Se a corrente detectar uma inconsistência entre o registro de listagem de um local de memória e o seu arquivo de transações, a corrente determina que os acessos de memória dentro da transação não são confiáveis e a transação deve ser tentada de novo. Além do mais, se a corrente tentar escrever em um local de memória dentro de uma transação e determinar que outra corrente já atualizou o local de memória dentro de outra transação incompleta, a primeira corrente pode aguardar ou tentar de novo a transação mais tarde, ou pode tentar resol-

ver a contenda com a outra corrente. Além disso, a corrente pode manter um registro de desfazer dos seus acessos de memória de escrever durante a transação, incluindo o valor original e o endereço do local, para permitir que a corrente desfaça essas operações, no caso em que as transações são abortadas.

Em algumas implementações, artigos manufaturados são proporcionados como produtos de programas de computadores. Uma implementação de um produto de programa de computador proporciona um meio de armazenamento de programa de computador, legível por um sistema de computador e codificando um programa de computador. Outra implementação de um programa de computador pode ser proporcionada em um sinal de dados de computador representado em uma onda portadora por um sistema de computação e codificando o programa de computador.

Outras implementações são também descritas e aqui relacionadas.

BREVES DESCRIÇÕES DOS DESENHOS

A Figura 1 ilustra duas correntes concorrentes acessando os locais de memória partilhada, por meio de interfaces de memórias transacionais de software de atualização direta exemplificativas.

A Figura 2 ilustra operações para executar um acesso de memória por uma interface de memória transacional de software de atualização direta exemplificativa.

A Figura 3 ilustra operações exemplificativas para listar uma operação de leitura em um local de memória dentro de uma transação.

A Figura 4 ilustra operações exemplificativas para listar uma operação de escrita em um local de memória dentro de uma transação.

A Figura 5 ilustra operações exemplificativas para praticar uma transação em uma interface de memória transacional de software de atualização direta exemplificativa.

A Figura 6 ilustra um sistema que pode ser útil na implementação da tecnologia descrita.

DESCRIÇÕES DETALHADAS

A Figura 1 ilustra duas correntes concorrentes 100 e 102 acessando locais de memória partilhada 104 e 106, por meio de interfaces de memórias transacionais de software de atualização direta exemplificativas 108 e 110 (por exemplo, interfaces de programação de aplicação ou APIs). A corrente 100 representa, por exemplo, uma computação de transferências de conta bancária (resultando no relatório 112), e a corrente 102 representa, por exemplo, uma computação de um total de duas contas bancárias (resultando no relatório 114). Os locais de memória partilhada 104 (A1) e 106 (A2) residem em uma região de memória partilhada 116 e armazenam os valores representando a quantidade de fundos em duas contas separadas, inicialmente $10 e $210, respectivamente.

Cada local de memória partilhada pode estar associado com uma gama de endereços de memória (por exemplo, um arranjo, uma lista, uma tabela, um objeto de software, etc.) dentro do sistema. Um registro de listagem, tal como e-rec 118 ou e-rec 120, age como uma estrutura de controle para acessos de memória transacional de software de atualização

direta, e está associado com cada local de memória. Um registro de listagem pode estar associado com um local de memória de vários modos, um registro de listagem para cada local de memória, um registro de listagem para cada objeto de software, ou um registro de listagem para um bloco de locais de memória, tal como uma página no espaço de endereço virtual do processo. Em uma implementação, um registro de listagem inclui dois campos: um campo de versão e um campo selecionado, embora o campo de versão possa também reter identificadores de transação. Em outra implementação, um registro de listagem inclui um único campo de versão, embora o campo de versão possa reter identificadores de transação ou ponteiros de arquivo de transações. Outras estruturas de registros de listagem são também consideradas.

No exemplo ilustrado, a corrente 100 inclui pelo menos duas operações atômicas em uma única transação, para transferência de $100 da primeira conta, correspondente ao endereço de memória 104, para a segunda conta, correspondente ao endereço de memória 106. Uma primeira operação adiciona 100 ao valor correspondente ao local de memória 104, e uma segunda operação subtrai 100 do valor correspondente ao local de memória 106, para fazer a transferência. A corrente 102 concorrente inclui uma operação atômica em uma transação, para totalizar as quantidades em ambas as contas.

Para garantir acessos de memória seguros para cada transação, as correntes 100 e 102 acessam os locais de memória partilhada 104 e 106, usando as APIs 108 e 110. Uma corrente executa as operações de leitura dentro de uma transa-

ção por uma API, diretamente em um local de memória, e ao final da transação, recebe um sinal indicando se a corrente pode estar segura dos resultados da transação. Igualmente, uma corrente executa as operações de escrita dentro de uma transação pela API, diretamente no local de memória partilhada, desde que outra corrente não tenha já marcado uma operação de escrita naquele local de memória partilhada.

Cada corrente mantém um arquivo de transações (por exemplo, arquivo de transações 122 ou arquivo de transações 124), para gerenciar os seus acessos de memória transacional de atualização direta. Em uma implementação, um arquivo de transações tem duas listas componentes, uma para as leituras e uma para as atualizações. Em uma implementação alternativa, um arquivo de transações tem uma única listagem, com as entradas das listas sendo parametrizadas como "leituras" ou "atualizações". Cada arquivo de transações também inclui um campo de estado, que retém um valor indicando ATIVO, ABORTADO OU COMPROMETIDO. As entradas das listas individuais incluem um ponteiro (por exemplo, uma referência) para um registro de listagem de um endereço de memória associado, e pode também incluir um valor de versão. Em uma implementação alternativa, uma entrada de listagem pode também incluir um campo selecionado. Outra estrutura de dados, chamada um registro de desfazer, retém uma lista de endereços de memória que tenham sido atualizados dentro da transação e os seus valores prévios (isto é, antes do local de memória ser atualizado). Em mais uma outra implementação, um registro único é usado, combinando as funções do arquivo de transações e do

arquivo de atualizações, com as entradas das listas sendo parametizadas como entradas de "leitura", "atualização" (ou escrita) ou "desfazimento".

Cada transação também mantém um arquivo de desfazimento (126 e 128), que pode ser usado para desfazer os acessos de escrita na memória de uma determinada transação. Em uma implementação, um arquivo de desfazimento registra o valor original e o endereço de memória associado com cada acesso de escrita da transação. Dessa maneira, se a transação for abortada, então as entradas no arquivo de desfazimento podem ser executadas em ordem reversa, para restaurar os locais de memória partilhada escritos pela transação aos seus estados pré-transação.

Seguindo-se o protocolo especificado para as APIs 108 e 110, as correntes 100 e 102 podem executar com segurança as seqüências de acessos de memória aos locais de memória partilhada dentro das transações individuais, sem o risco de interferência não detectada por outra corrente.

A Figura 2 ilustra as operações 200 para execução de uma seqüência de acessos de memória dentro de uma transação, por meio de uma interface de memória transacional de software de atualização direta exemplificativa. As operações 200 representam as chamadas feitas pela API por uma corrente individual, acessando um ou mais locais de memória partilhada. Por exemplo, um local de memória partilhada pode corresponder a um objeto de software para uma conta bancária individual. Cada local de memória partilhada é associado com um endereço de memória e um registro de listagem, na criação do

correspondente objeto de software.

Uma operação de iniciação 202 inclui uma nova transação para uma corrente. Em uma implementação, a operação de iniciação 202 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação "TransactionStart()", que aloca e inicializa um arquivo de transações para a nova transação. O arquivo de transações é mantido no local memória (isto é, não partilhada) acessível pela corrente. Na iniciação, o arquivo de transações está vazio e o seu estado é ajustado para ATIVO. Em uma implementação, a operação de iniciação 202 também aloca e inicializa um novo arquivo de desfazimento associado com a transação.

Uma operação de listagem 204 lista um local de memória específico para uma operação de acesso de memória, tal como uma operação de leitura ou uma operação de escrita. Em uma implementação, a operação de listagem 204 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação "EnlistAddrForRead(Addr r)", em que o endereço r representa o endereço do local de memória específico. Em outra implementação, a operação de listagem 204 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação "EnlistAddrForWrite(Addr r)", em que o endereço r representa o endereço do local de memória específico. Uma descrição mais detalhada de operações de listagem é proporcionada abaixo.

Uma operação de acesso 206 executa um acesso de

memória para o local de memória específico, tal como por uma operação de leitura ou uma operação de escrita. Em uma implementação, a operação de acesso 206 é representada por uma leitura direta no local de memória específico. Em outra implementação, a operação de acesso 206 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação de "TransactionRead(Addr r)", em que o endereço r representa o endereço do local de memória específico e a operação lê o valor diretamente do local de memória específico. Em mais uma outra implementação, a operação de acesso 206 é representada por uma escrita direta no local de memória específico. Nessa implementação, uma operação UpdateIndo(Addr r) pode ser também empregada para atualizar um arquivo de desfazimento (como discutido acima). Em mais uma outra implementação, a operação de acesso 206 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação "TransactionWrite(Addr r, Value v)", em que o endereço r representa o endereço do local de memória específico, o valor v representa o valor a ser escrito no local de memória, e a operação adiciona uma entrada a um arquivo de desfazimento.

Uma operação de decisão 208 determina se outra operação de acesso de memória se mantém na seqüência de transação. Sendo assim, e se a transação não tiver acessado previamente o local de memória específico da operação de acesso de memória seguinte, o processamento prossegue para a operação de listagem 204. Alternativamente, o processamento pode

evitar a operação de listagem 204, porque o local de memória específico já tinha sido listado dentro da transação atual.

Em qualquer caso, as etapas de processamento por cada acesso de memória na seqüência de transação, até não mais operações de acesso de memória permanecerem na seqüência de transação, em cujo momento uma operação de execução 210 tenta executar a transação. Em uma implementação, a operação de execução 210 é representada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação "TransactionCommit()". Geralmente, a operação de execução 210 determina se todas as operações de leitura de memória dentro da transação foram completadas dentro da transação, sem uma atualização de interferência por outra corrente. Se a operação de execução 210 falha, a corrente determina que a transação falhou e vai reintroduzir toda a transação, se adequado. Se a operação de execução 210 é bem-sucedida, então a corrente determina que a transação foi completada com sucesso e que a corrente pode confiar nos resultados da transação (por exemplo, os valores lidos e as operações de atualização). Ao final, a operação de execução 210 "retira" as listagens da transação, de modo que os locais de memória associados ficam acessíveis a outras correntes. Em uma implementação, a retirada é feita por iteração pelas entradas de transação de escrita no arquivo de transações e, para cada registro de listagem associado, armazenamento de um novo número de versão no campo de versão do registro de listagem (por exemplo, incrementação do número de versão) de um local de memória atualizado.

É possível exceder os números de versões (por exemplo, todos os números disponíveis no campo de versão podem ter sido usados com o tempo). Por exemplo, se a versão for incrementada por um com cada atualização (por exemplo, 0, 1, 2, 3, etc.) e exceder, eventualmente, um número de versão máximo "N", então um mecanismo para "derrubar" ou de outro modo tratar o transbordamento do número de versão pode ficar em ordem. Em uma implementação, se o número de transações executadas com o tempo exceder N, então as transações ativas nesse ponto podem ser abortadas. Alternativamente, se o número de transações executadas com o tempo exceder N, as transações ativas nesse ponto podem ser parcialmente executadas, de modo que transações inválidas são abortadas e transações válidas se mantêm ativas. Em mais uma outra interface de memória transacional de software de atualização direta exemplificativa, o tamanho N pode ser aumentado de modo que os processos aceitáveis sejam extremamente improváveis de exceder N.

Uma operação de aborto, que pode ser proporcionada dentro de uma interface de memória transacional de software de atualização direta exemplificativa por uma operação TransactionAbort(), abandona a transação atual associada com a atual corrente e recupera quaisquer atualizações feitas dentro da transação, antes do evento de aborto. Em uma implementação, o campo de estado no arquivo de transações atual é ajustado para ABORTADO, e as entradas no arquivo de desfazimento são executadas ao contrário, restaurando os valores pré-transação nos locais de memória atualizados. Uma vez que

as atualizações da transação abortadas são desfeitas, a operação de aborto "abandona" as listagens.

A Figura 3 ilustra as operações exemplificativas 300 para listar uma operação de leitura em um local de memória dentro de uma transação. Responsiva a uma chamada por uma corrente a uma operação de leitura de listagem (consultar discussão relativa à Figura 2), relativa a um determinado endereço de memória r, uma operação de localização 302 localiza o registro de listagem (isto é, "e-rec") associado com o endereço de memória r. Em uma implementação, o registro de listagem é uma estrutura de dados associada com um endereço de memória r (por exemplo, como parte de um objeto de software no endereço de memória r, ou como referido por aquele objeto de software).

Uma operação de versão 304 lê o valor de versão associado com o endereço de memória do registro de listagem localizado. Uma operação de decisão 306 determina se o valor de versão de leitura representa efetivamente uma versão, um identificador indicando a transação atual, ou um identificador indicando uma outra transação. Se o valor de versão de leitura representar uma versão, então uma operação de arquivamento 308 adiciona a versão e um ponteiro (por exemplo, uma referência), para registro de listagem localizado, a uma entrada em uma lista de transações de leitura (por exemplo, uma lista ligada) do arquivo de transações. A corrente pode então prosseguir para ler o valor diretamente no endereço de memória.

Se o valor de versão de leitura representar um i-

dentificador escrito no registro de listagem do endereço de memória pela transação atual, então o arquivo de transações já tinha sido atualizado por uma operação de escrita na transação atual, e a corrente pode meramente continuar (como indicado pela operação de confirmação 310) para ler o valor diretamente no endereço de memória. Em continuação, a corrente tem a confirmação de que uma entrada de arquivo de transações para o registro de listagem foi gerada no arquivo de transações da transação. Um identificador de uma transação pode ser um identificador de transação único, arbitrário, um ponteiro (por exemplo, uma referência) de volta para um campo selecionado no arquivo de transações, ou algum outro identificador único.

Se o valor de versão de leitura representar um identificador escrito no registro de listagem do endereço de memória por outra transação, então o arquivo de transações já foi atualizado por uma operação de escrita na outra transação, e a corrente invoca uma operação de resolução 312 e depois tenta novamente a listagem. Em uma identificador, uma operação de resolução exemplificativa 312 é efetivamente uma instrução não operacional, que faria com que a operação de listagem girasse e aguardasse. Alternativamente, uma aparência de exclusão normal poderia ser associada com o registro de listagem para evitar giro. Em mais uma outra implementação alternativa, cada arquivo de transações pode ser associado com uma aparência revogável. Nessa implementação, a operação de resolução de uma transação de acesso posterior (T2) pode adquirir a aparência revogável da transação anterior

(T1). Com a aparência revogável, T2 pode executar uma operação TransactionAbort() em nome de T1, e depois liberar a aparência revogável de T1. Depois, T2 pode relistar o endereço de memória r para sua operação de leitura, e T1 pode ser deslocado para uma função de recuperação que pode promover uma exceção para indicar que T1 foi abortada e deve ser reiniciada em um ponto adequado no processamento.

Em uma implementação alternativa, uma operação EnlistAddrForRead() omite a operação de decisão 306 do valor de versão que foi lido e tão logo assume o conteúdo do campo de versão no registro de listagem é uma versão. Nessa implementação, as operações TransactionCommit() captura quaisquer conflitos em um local de memória partilhada, se o conflito for provocado por um conflito de versão ou por reserva do local de memória partilhada por um acesso de escrita de outra corrente.

A Figura 4 ilustra as operações exemplificativas 400 para listar uma operação de escrita em um local de memória dentro de uma transação. Responsiva a uma chamada por uma corrente para uma operação de escrita de listagem (consultar a discussão relativa à Figura 2), relativa a um determinado endereço de memória r, uma operação de localização 402 localiza o registro de listagem (isto é, "e-rec") associado com o endereço de memória r. Em uma implementação, o registro de listagem é uma estrutura de dados associada com um endereço de memória r (por exemplo, como parte de um objeto de software no endereço de memória r, ou como referido por aquele objeto de software).

Uma operação de versão 404 lê o valor de versão associado com o endereço de memória do registro de listagem localizado. Uma operação de decisão 406 determina se o valor de versão de leitura representa efetivamente uma versão, um identificador indicando a transação atual, ou um identificador indicando uma outra transação. Se o valor de versão lido representar uma versão, então uma operação de arquivamento 408 adiciona a versão e um ponteiro (por exemplo, uma referência), para registro de listagem localizado, a uma entrada em uma lista de transações de escrita (por exemplo, uma lista ligada) do arquivo de transações. Uma operação de marcação 410 usa uma operação de comparação e troca atômica para ajustar uma marca designando uma transação atual no campo de versão do registro de listagem do endereço de memória r, desse modo, reservando temporariamente o endereço de memória r (e o local de memória associado, a gama de locais de memória, objeto, etc.) para as transações atuais. Se a operação de comparação e troca é bem-sucedida (determinada pela operação de decisão 412), o valor de versão salvo na lista de transações de escrita dentro do arquivo de transações é escrito em um campo selecionado do registro de listagem, e depois a corrente pode então prosseguir para atualizar o valor diretamente no endereço de memória. Alternativamente, se a operação de comparação e troca falha, a versão e o ponteiro (por exemplo, uma referência) para o registro de listagem localizado são apagados da lista de transações de escrita dentro do arquivo de transações e a listagem é de novo tentada na operação de localização 402.

Em uma implementação, o campo selecionado pode ser incluído em uma entrada de transação, em vez de um registro de listagem. Nessa implementação, um identificador de transação em um campo de versão do registro de listagem é um ponteiro (por exemplo, uma referência) para a entrada de transação. Dessa maneira, podem ser empregadas operações de comparação e troca rápidas, e a versão selecionada é armazenada na memória não partilhada local, apenas para os locais de memória atualizados em uma transação.

Se o valor da versão de leitura representar um identificador escrito no registro de listagem do endereço de memória pela transação atual, então o arquivo de transações já tinha sido atualizado por uma operação de escrita na transação atual, e a corrente pode meramente continuar (como indicado pela operação de continuação 416), para atualizar o valor diretamente no endereço de memória. Se o valor de versão de leitura representar um identificador escrito para o registro de listagem do endereço de memória por outra transação, então o arquivo de transações já tinha sido atualizado por uma operação de escrita na outra transação. Como tal, a corrente invoca uma operação de resolução 418 e depois tenta novamente a listagem. As operações de resolução exemplificativas já foram discutidas com relação à operação de resolução 418.

A Figura 5 ilustra operações exemplificativas 500 para executar uma transação em uma interface de memória transacional de software de atualização direta. Uma operação de entrada de arquivo 502 lê o ponteiro (por exemplo, uma

referência) de uma entrada na lista de transações de leitura do arquivo de transações. Usando esse ponteiro, uma operação de registro de listagem 504 lê o campo de versão do registro de listagem associado com o endereço de memória. Uma operação de decisão 506 determina se o valor lido do campo de versão realmente representa uma versão, uma marca indicando a transação atual, ou uma marca indicando outra transação.

Se o valor lido do campo de versão for de fato uma versão, uma operação de leitura 508 lê a versão da entrada do arquivo de transações, do qual o ponteiro (por exemplo, uma referência) foi lido na operação de entrada de arquivos 502. Se as duas versões são iguais (determinada pela operação de decisão 510), a corrente pode ser ainda capaz de ser completada com sucesso, porque o endereço de memória associado com a entrada de arquivo atual não foi atualizada por outra corrente, durante a transação. Se isso for verdade para todas as entradas de arquivos na lista de transações de leitura, todas as leituras dentro da atual transação para aquele endereço de memória são válidas e a transação pode ser válida.

Uma operação de decisão 512 determina se uma entrada de arquivo de transação seguinte existe na lista de transações de leitura. Sendo assim, o processamento prossegue para a operação de entrada de arquivos 502. De outro modo, uma operação de sucesso 514 sinaliza um sucesso da operação de execução e ajusta o estado da transação atual para EXECUTADO. Uma operação de versão 520 atualiza os valores de versão de todos os registros de listagem associados com as

entradas das listas de transações de escrita, para indicar uma transação completada com sucesso em associação com o local de memória. Uma operação de terminação 522 remove todas as entradas de arquivos de transações dos arquivos de transações e elimina o próprio arquivo de transações.

Se a operação de decisão 510 determinar que as versão não são iguais, a incompatibilidade indica que as leituras para o local de memória não são válidas, porque o endereço de memória foi atualizado por outra corrente durante a transação atual. Nesse caso, o processamento prossegue para uma operação de falha 518, que sinaliza uma falha da operação de execução. Além do mais, uma operação de aborto aborta a transação, como descrito acima.

Se a operação de decisão 506 determinar que o valor lido do campo de versão do registro de listagem representa de fato um identificador, indicando a transação atual, uma operação de leitura 512 lê um valor de versão de um campo selecionado. Em uma implementação, o campo selecionado é parte do registro de listagem do endereço de memória. Em uma implementação alternativa, o campo selecionado é armazenado em uma entrada no arquivo de transações. O processamento então prossegue para a operação de leitura 508 e a operação de decisão 510, para comparar a versão do campo selecionado para a versão do arquivo de transações.

Se a operação de decisão 506 determinar que o valor lido do campo de versão do registro de listagem representar de fato um identificador, indicando outra transação, a atualização para o endereço de memória é impedida por uma

atualização de outra transação concorrente. Portanto, a operação de falha 518 sinaliza uma falha de transação. Além do mais, uma operação de aborto aborta a transação, como descrito acima.

O hardware exemplificativo e o meio físico operacional da Figura 6, para implementar a invenção, incluem um dispositivo de computação multipropósito na forma de um computador 20, incluindo uma unidade de processamento 21, uma memória de sistema 22, e um barramento de sistema 23, que acopla operacionalmente os vários componentes do sistema, incluindo a memória de sistema à unidade de processamento 21. Pode haver apenas uma ou pode haver mais de uma unidade de processamento 21, de modo que o processador do computador 20 compreende uma única unidade de processamento central (CPU), ou uma pluralidade de unidades de processamento, referidas comumente como um meio físico de processamento paralelo. O computador 20 pode ser um computador convencional, um computador distribuído ou qualquer outro tipo de computador, a invenção não sendo assim limitada.

O barramento de sistema 23 pode ser qualquer um de vários tipos de estruturas de barramento, incluindo um barramento de memória ou um controlador de memória, um barramento periférico, uma estrutura comutada, uma conexão ponto a ponto, e um barramento local usando qualquer uma de várias arquiteturas de barramento. A memória de sistema pode ser também referida simplesmente como memória, e inclui memória exclusiva de leitura (ROM) 24 e memória de acesso aleatório (RAM) 25. Um sistema básico de entrada / saída (BIOS) 26,

contendo as rotinas básicas que ajudam a transferir informações entre os elementos dentro do computador 20, tal como durante partida, é armazenado na ROM 24. O computador 20 inclui ainda uma unidade de disco rígido 27, para leitura de e escrita em um disco rígido, não mostrado, uma unidade de disco magnético 28, para leitura de ou escrita em um disco magnético removível 29, e uma unidade de disco óptico 30, para leitura de ou escrita em um disco óptico removível 31, tal como um CD ROM ou outra mídia óptica.

A unidade de disco rígido 27, a unidade de disco magnético 28 e a unidade de disco óptico 30 são ligadas ao barramento de sistema 23 por uma interface de unidade de disco rígido 32, uma interface de unidade de disco magnético 33 e uma interface de unidade de disco óptico 34, respectivamente. As unidades e os seus meios legíveis por computador associados proporcionam armazenamento não volátil de instruções legíveis por computador, estruturas de dados, módulos de programas e outros dados para o computador 20. Aqueles versados na técnica vão considerar que qualquer tipo de meio legível por computador, que pode armazenar dados, que seja acessível por um computador, tais como cassetes magnéticos, cartões de memória instantânea, videodiscos digitais, memórias de acesso aleatório (RAMs), memórias exclusivas de leitura (ROMs), e assemelhados, pode ser usado no meio físico operacional exemplificativo.

Vários módulos de programas podem ser armazenados no disco rígido, disco magnético 29, disco óptico 31, ROM 24 ou RAM 25, incluindo um sistema operacional 35, um ou mais

programas de aplicação 36, outros módulos de programas 37, e dados de programas 38. Um usuário pode introduzir comandos e informações no computador pessoal 20 pelos dispositivos de entrada, tais como um teclado 40 e um dispositivo apontador 42. Outros dispositivos de entrada (não mostrados) podem incluir um microfone, um joystick, um acionador de jogo, uma antena parabólica, um escâner, ou assemelhados. Esses e outros dispositivos de entrada são freqüentemente conectados à unidade de processamento 21 por uma interface de porta serial 46, que é acoplada ao barramento do sistema, mas podem ser conectados por outras interfaces, tal como uma porta paralela, uma porta de jogo ou um barramento serial universal (USB). Um monitor 47 ou outro tipo de dispositivo visor é também conectado ao barramento do sistema 23 por uma interface, tal como um adaptador de vídeo 48. Além do monitor, os computadores incluem tipicamente outros dispositivos de saída periféricos (não mostrados), tais como alto-falantes e impressoras.

O computador 20 pode operar em um meio físico ligado em rede, usando conexões lógicas a um ou mais computadores remotos, tal como o computador remoto 49. Essas conexões lógicas são feitas por um dispositivo de comunicação acoplado ao, ou a uma parte do, computador 20; a invenção não sendo limitada a um tipo particular de dispositivo de comunicações. O computador remoto 49 pode ser outro computador, um servidor, um roteador, um PC em rede, um cliente, um dispositivo local ou outro nó de rede comum, e inclui, tipicamente, muitos ou todos dos elementos descritos acima rela-

tivos ao computador 20, embora apenas um dispositivo de armazenamento de memória 50 tenha sido ilustrado na Figura 6. As conexões lógicas ilustradas na Figura 6 incluem uma rede de área local (LAN) 51 e uma rede de longa distância (WAN) 52. Esses meios físicos ligados em rede são usuais em redes de escritórios, redes de computadores de empresas, redes internas e a Internet, que são todos os tipos de redes.

Quando usado em um meio físico ligado em rede - LAN, o computador 20 é conectado à rede local 51 por uma interface ou adaptador de rede 53, que é um tipo de dispositivo de comunicação. Quando usado em um meio físico ligado em rede - WAN, o computador 20 inclui, tipicamente, um modem 54, um adaptador de rede, um tipo de dispositivo de comunicação, ou qualquer outro tipo dispositivo de comunicações, para estabelecer comunicações pela rede de longa distância 52. O modem 54, que pode ser interno ou externo, é conectado ao barramento do sistema 23 pela interface de porta serial 46. Em um meio físico ligado em rede, os módulos de programas ilustrados relativos ao computador pessoal 20, ou partes dele, podem ser armazenados no dispositivo de armazenamento de memória remoto. Considera-se que as conexões de rede mostradas são exemplificativas, e outros meios de dispositivos de comunicações, para estabelecer uma ligação de comunicações entre os computadores, podem ser usados.

Em uma implementação exemplificativa, um módulo de TransactionStar(), um módulo de EnlistAddrForRead, um módulo de EnlistAddrForWrite(), um módulo de TransactionRead(), um módulo de TransactionAbort(), um módulo de TransactionCom-

mit() e outros módulos podem ser incorporados como parte do sistema operacional 35, programas de aplicação 36, ou outros módulos de programas 37. Os arquivo de transações, os registros de listagem e outros dados podem ser armazenados como dados de programas 38.

A tecnologia descrita aqui é implementada como operações e/ou módulos lógicos em um ou mais sistemas. As operações lógicas podem ser implementadas (1) como uma seqüência de etapas implementadas por processador, executadas em um ou mais sistemas de computadores, e (2) como módulos de máquinas ou circuitos interligados dentro de um ou mais sistemas de computação. Igualmente, as descrições de vários módulos componentes podem ser proporcionadas em termos de operações executadas ou feitas pelos módulos. A implementação resultante é uma questão de seleção, dependente dos requisitos de desempenho do sistema básico implementando a tecnologia descrita. Conseqüentemente, as operações lógicas, constituindo as modalidades da tecnologia aqui descrita, são aqui referidas variadamente como operações, etapas, objetos ou módulos. Além do mais, deve-se entender que as operações lógicas podem ser conduzidas em qualquer ordem, a menos que reivindicado explicitamente de outro modo, ou uma ordem específica é inerentemente necessária pela linguagem da reivindicação.

O relatório descrito, os exemplos e os dados apresentados acima proporcionam uma descrição completa da estrutura e do uso das modalidades exemplificativas da invenção. Uma vez que muitas modalidades da invenção podem ser elabo-

radas sem afastamento do espírito e do âmbito da invenção, a invenção reside nas reivindicações anexadas a seguir. Em particular, deve-se entender que a tecnologia descrita pode ser empregada independente de um computador pessoal. Outras modalidades são, portanto, consideradas.

PI0613946-9

REIVINDICAÇÕES

1. Método de gerenciamento de acesso de memória a um local de memória partilhada dentro de uma transação de uma primeira corrente, o local de memória partilhada sendo acessível pela primeira corrente e uma segunda corrente, o método **CARACTERIZADO** pelo fato de que compreende:

identificar um registro de listagem associado com o local de memória partilhada;

marcar o registro de listagem para reservar o local de memória partilhada para a transação, se o registro de listagem não indicar nenhum conflito com um acesso de memória de escrita ao local de memória partilhada pela segunda corrente; e

registro de uma referência ao local de memória partilhada e do seu conteúdo em um arquivo de desfazimento.

2. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que compreende ainda executar dentro da transação um acesso de memória de escrita ao local de memória partilhada, responsivo à operação de marcação.

3. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que a operação de marcação compreende escrever um identificador de transação no registro de listagem, o identificador de transação identificando a transação da primeira corrente.

4. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que o registro de listagem indica um conflito por um acesso de memória de escrita ao local de memória partilhada por uma segunda corrente, se o registro

de listagem for marcado pela segunda corrente.

5. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que o arquivo de desfazimento inclui uma seqüência ordenada de referências e conteúdo associado de local de memória partilhada, e compreendendo ainda escrever o conteúdo nos locais de memória partilhada associados, em que os acessos de escrita individuais são feitos em ordem reversa na seqüência.

6. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que compreende ainda registrar uma entrada de arquivo de transações, incluindo uma referência ao registro de listagem, a entrada do arquivo de transações sendo registrada em um arquivo de transações contendo entradas de arquivo de transações de uma pluralidade de acessos de memória de escrita dentro da transação.

7. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que compreende ainda, se o registro de listagem indicar um conflito por um acesso de memória de escrita ao local de memória partilhada pela segunda corrente, resolver o conflito para permitir um acesso de memória de escrita ao local de memória partilhada, dentro da transação da primeira corrente.

8. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que compreende ainda escrever um novo valor de versão no registro de listagem do local de memória partilhada ao final da transação.

9. Método, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que o registro de listagem inclui

um campo de versão, e a operação de marcação compreende:

registrar o conteúdo do campo de versão em um campo selecionado; e

escrever um identificador de transação no campo de versão.

10. Meio legível por computador **CARACTERIZADO** pelo fato de que tem instruções executáveis por computador, para executar um processo de computador para implementação do método do tipo definido na reivindicação 1.

11. Método de gerenciamento de acesso de memória a um local de memória partilhada dentro de uma transação de uma primeira corrente, o local de memória partilhada sendo acessível pela primeira corrente e uma segunda corrente, o método **CARACTERIZADO** pelo fato de que compreende:

identificar um registro de listagem associado com o local de memória partilhada;

gerar uma entrada de arquivo de transações associada com a transação, a entrada de arquivo de transações fazendo referência ao registro de listagem associado com o local de memória partilhada;

executar o acesso de memória de leitura ao local de memória partilhada dentro da transação pela primeira corrente; e

executar a transação para o acesso de memória de leitura, se a entrada de arquivo de transações e o registro de listagem referido não indicarem um conflito por um acesso de memória de escrita no local de memória partilhada pela segunda corrente, responsivo à operação de execução.

12. Método, de acordo com a reivindicação 11, **CARACTERIZADO** pelo fato de que o registro de listagem inclui um campo de versão, e compreende ainda:

ler o conteúdo do campo de versão do registro de listagem, antes da operação de execução; e

se o conteúdo representar um identificador de transação da primeira corrente, acusar que a entrada do arquivo de transações para o registro de listagem já foi gerada dentro da transação.

13. Método, de acordo com a reivindicação 11, **CARACTERIZADO** pelo fato de que o registro de listagem inclui um campo de versão, e compreendendo ainda:

ler o conteúdo do campo de versão do registro de listagem, antes da operação de execução; e

se o conteúdo representar um identificador de transação da segunda corrente, abortar a transação.

14. Método, de acordo com a reivindicação 11, **CARACTERIZADO** pelo fato de que o registro de listagem inclui um campo de versão, e compreendendo ainda:

ler o conteúdo do campo de versão do registro de listagem, antes da operação de execução; e

se o conteúdo representar um identificador de transação, registrar o valor de versão na entrada do arquivo de transações.

15. Método, de acordo com a reivindicação 11, **CARACTERIZADO** pelo fato de que compreende ainda identificar um conflito por um acesso de memória de escrita ao local de memória partilhada pela segunda corrente, se um valor de

versão registrado na entrada do arquivo de transações não for igual a um valor de versão registrado no registro de listagem, quando da execução.

16. Método, de acordo com a reivindicação 15, **CARACTERIZADO** pelo fato de que compreende ainda abortar a transação, se um conflito for identificado.

17. Método, de acordo com a reivindicação 11, **CARACTERIZADO** pelo fato de que o registro de listagem inclui um campo de versão, e compreende ainda:

ler de um campo selecionado associado com o registro de listagem, um valor de versão previamente armazenado no campo de versão do registro de listagem; e

identificar um conflito por um acesso de memória de escrita ao local de memória partilhada pela segunda corrente, se um valor de versão registrado na entrada do arquivo de transações não for igual ao valor de versão registrado no campo selecionado, quando da execução.

18. Meio legível por computador **CARACTERIZADO** pelo fato de que tem instruções executáveis por computador, para executar um processo de computador para implementação do método tipo definido na reivindicação 11.

19. Método de gerenciamento de um acesso de memória a locais de memória partilhada dentro de uma transação de uma primeira corrente, os locais de memória partilhada sendo acessíveis pela primeira corrente e uma segunda corrente, o método **CARACTERIZADO** pelo fato de que compreende:

listar os locais de memória partilhada dentro da transação;

manter um arquivo de transações, incluindo uma entrada de arquivo de transações associada com a transação e um registro de listagem para cada local de memória partilhada;

manter um arquivo de desfazimento, incluindo entradas de acessos de memória de escrita aos locais de memória partilhada, cada entrada incluindo um endereço associado com um local de memória partilhada do acesso de memória de escrita e conteúdo anterior do local de memória partilhada;

executar acessos de memória diretamente nos locais de memória partilhada; e

realizar a transação para os acessos de memória, responsiva à operação de execução, se nenhum conflito por um acesso de memória de escrita ao local de memória partilhada pela segunda corrente for detectado.

20. Meio legível por computador **CARACTERIZADO** pelo fato de que tem instruções executáveis por computador, para executar um processo de computador para implementação do método tipo definido na reivindicação 19.

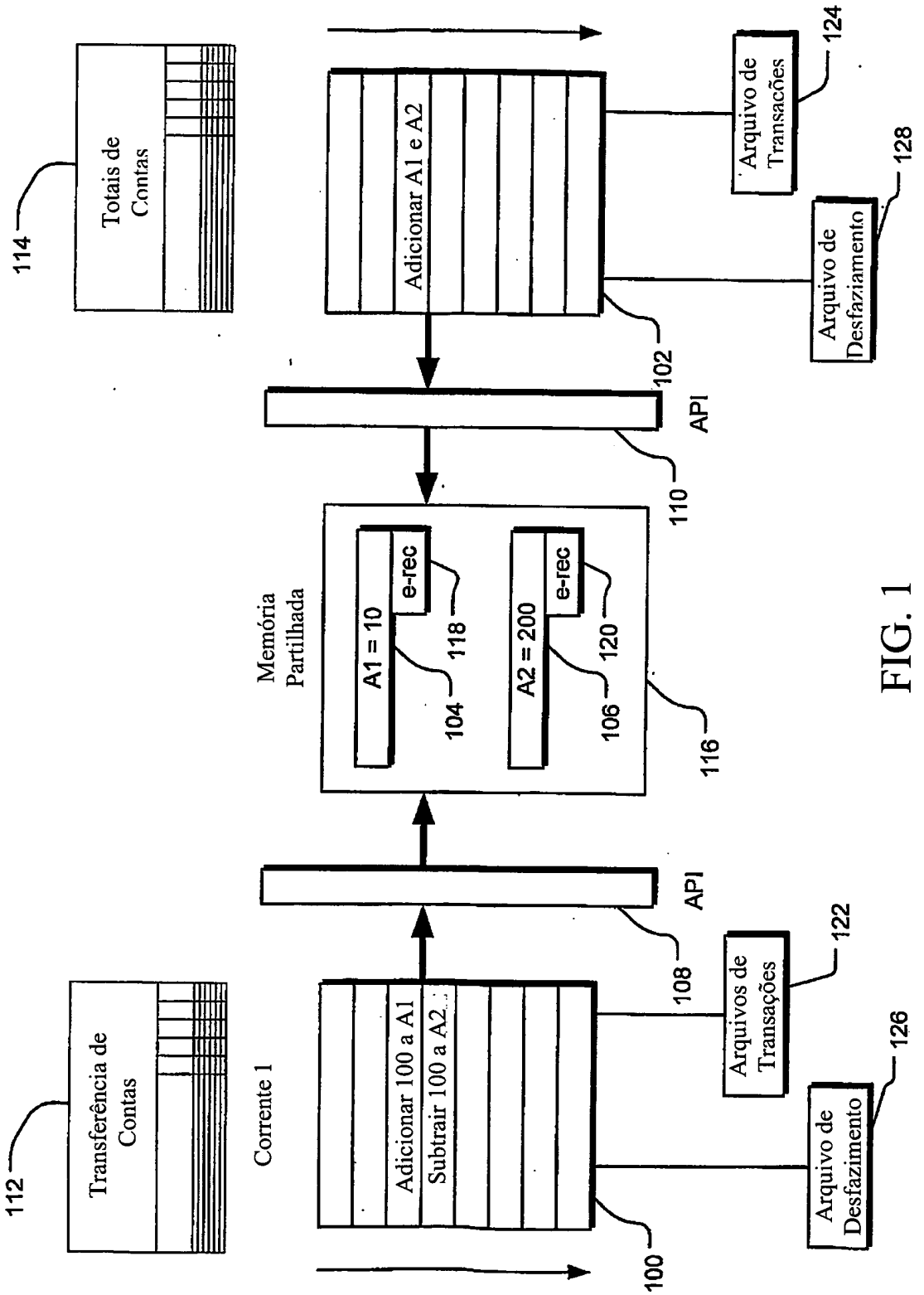

Totais de Contas
114
Adicionar A1 e A2
Arquivo de Transacões
124
Arquivo de Desfaziamento
128
102
API
110
Memória Partilhada
A1 = 10
e-rec
118
104
A2 = 200
e-rec
120
106
116
API
108
Arquivos de Transações
122
Arquivo de Desfazimento
126
Corrente 1
Adicionar 100 a A1
Subtrair 100 a A2
100
Transferência de Contas
112

FIG. 1

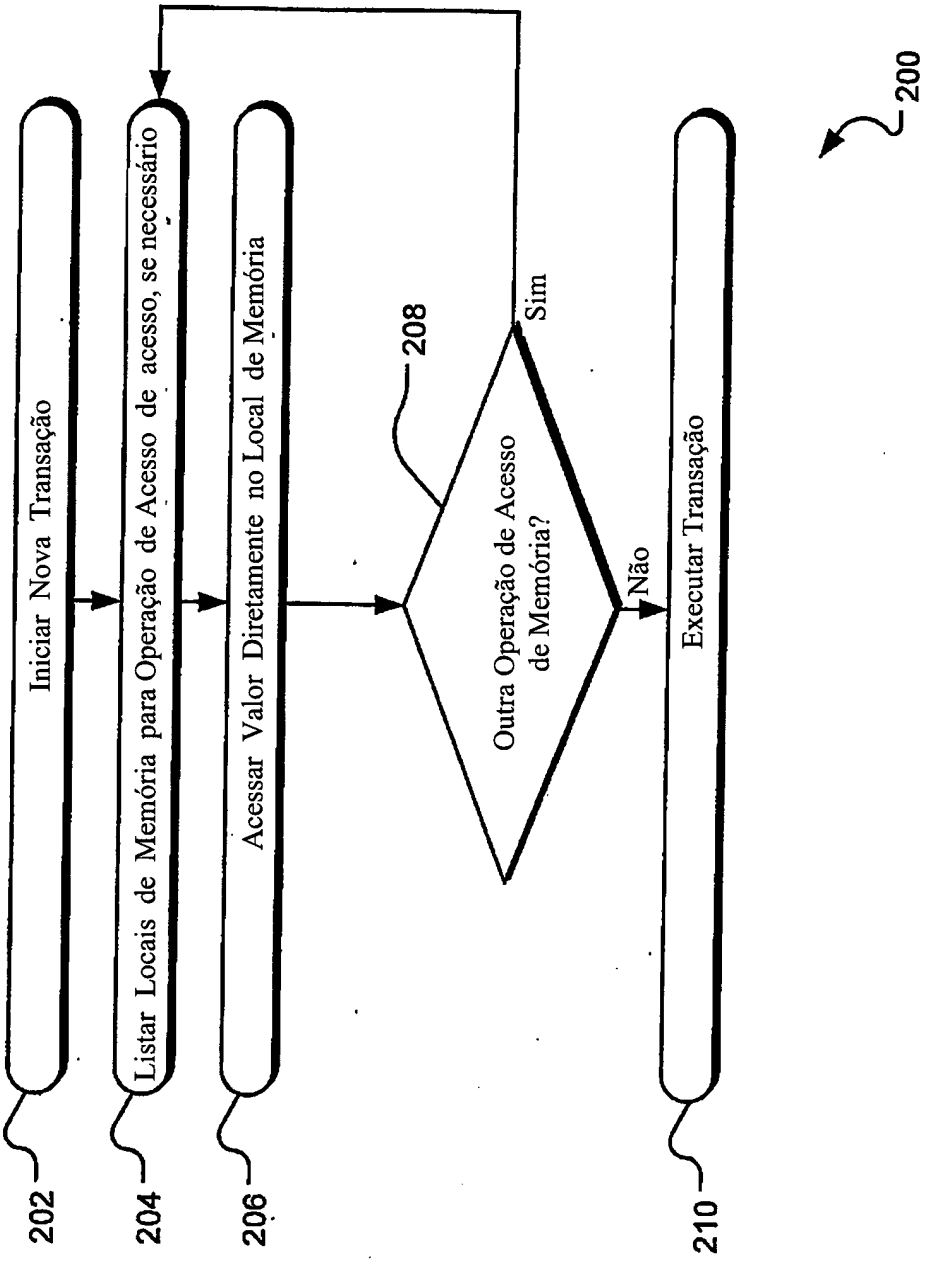

Iniciar Nova Transação
Listar Locais de Memória para Operação de Acesso de acesso, se necessário
Acessar Valor Diretamente no Local de Memória
Outra Operação de Acesso de Memória?
Sim
Não
Executar Transação
202
204
206
208
210
200

FIG. 2

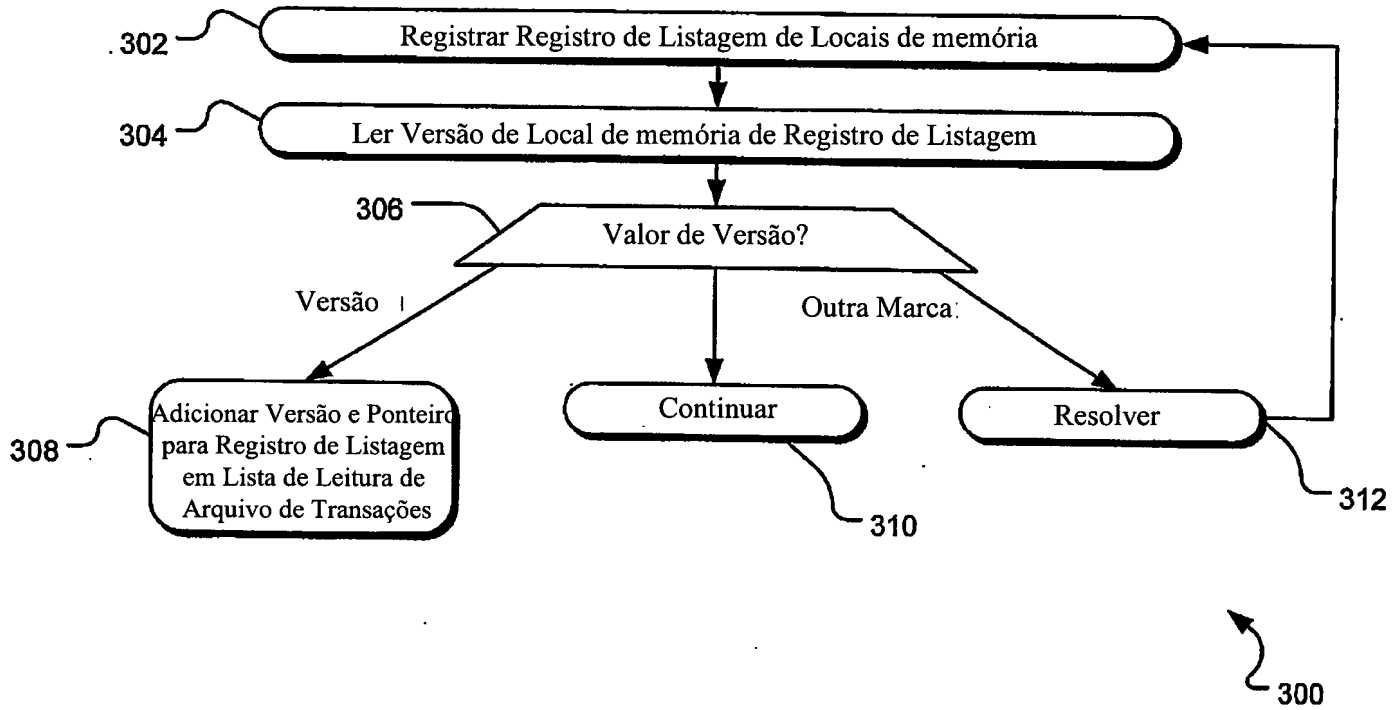

302
Registrar Registro de Listagem de Locais de memória
304
Ler Versão de Local de memória de Registro de Listagem
306
Valor de Versão?
Versão
Outra Marca
308
Adicionar Versão e Ponteiro para Registro de Listagem em Lista de Leitura de Arquivo de Transações
Continuar
310
Resolver
312
300

FIG. 3

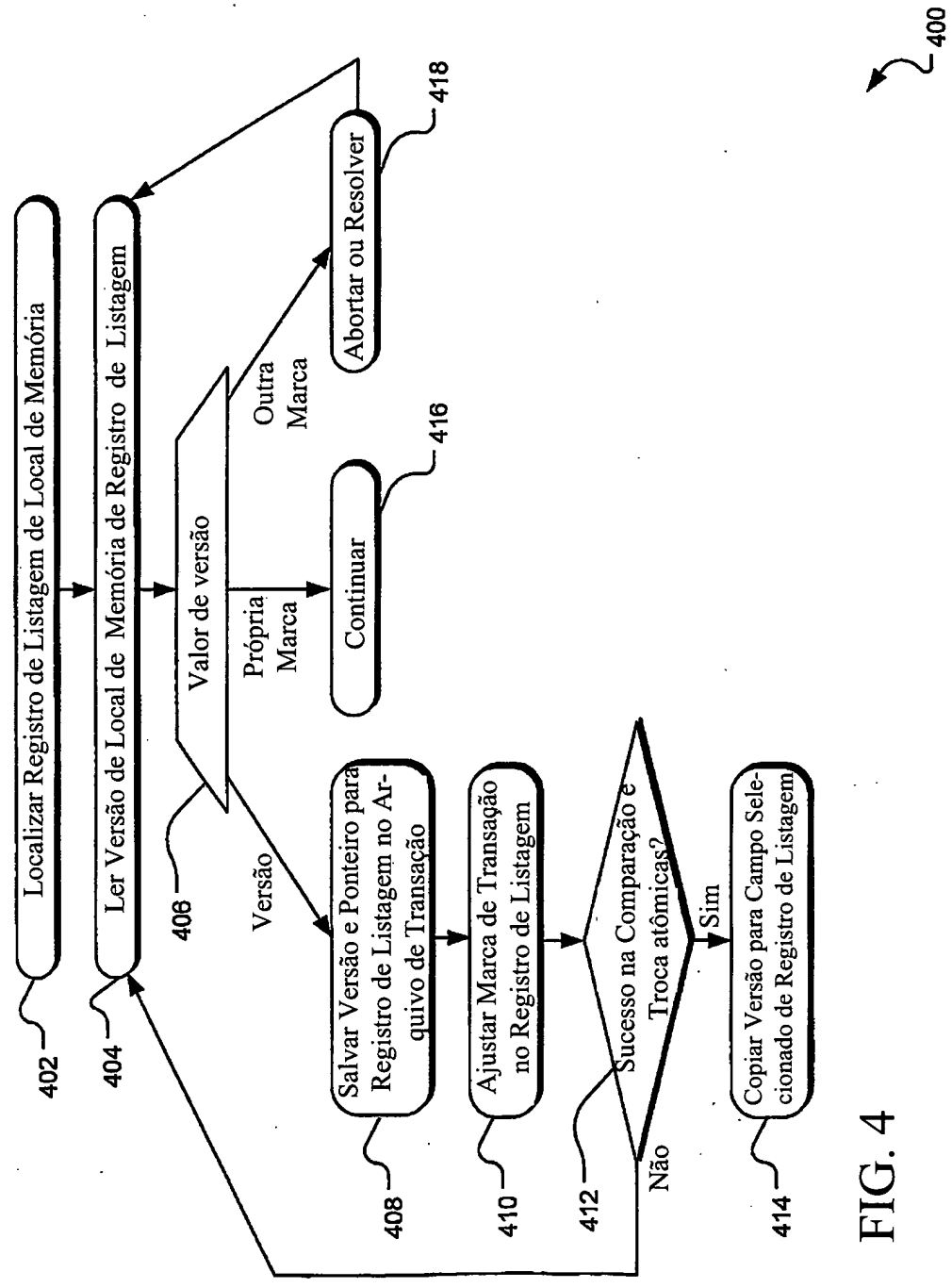

Localizar Registro de Listagem de Local de Memória
Ler Versão de Local de Memória de Registro de Listagem
Valor de versão
Própria Marca
Outra Marca
Continuar
Abortar ou Resolver
Versão
Salvar Versão e Ponteiro para Registro de Listagem no Arquivo de Transação
Ajustar Marca de Transação no Registro de Listagem
Sucesso na Comparação e Troca atômicas?
Não
Sim
Copiar Versão para Campo Selecionado de Registro de Listagem
402
404
406
408
410
412
414
416
418
400

FIG. 4

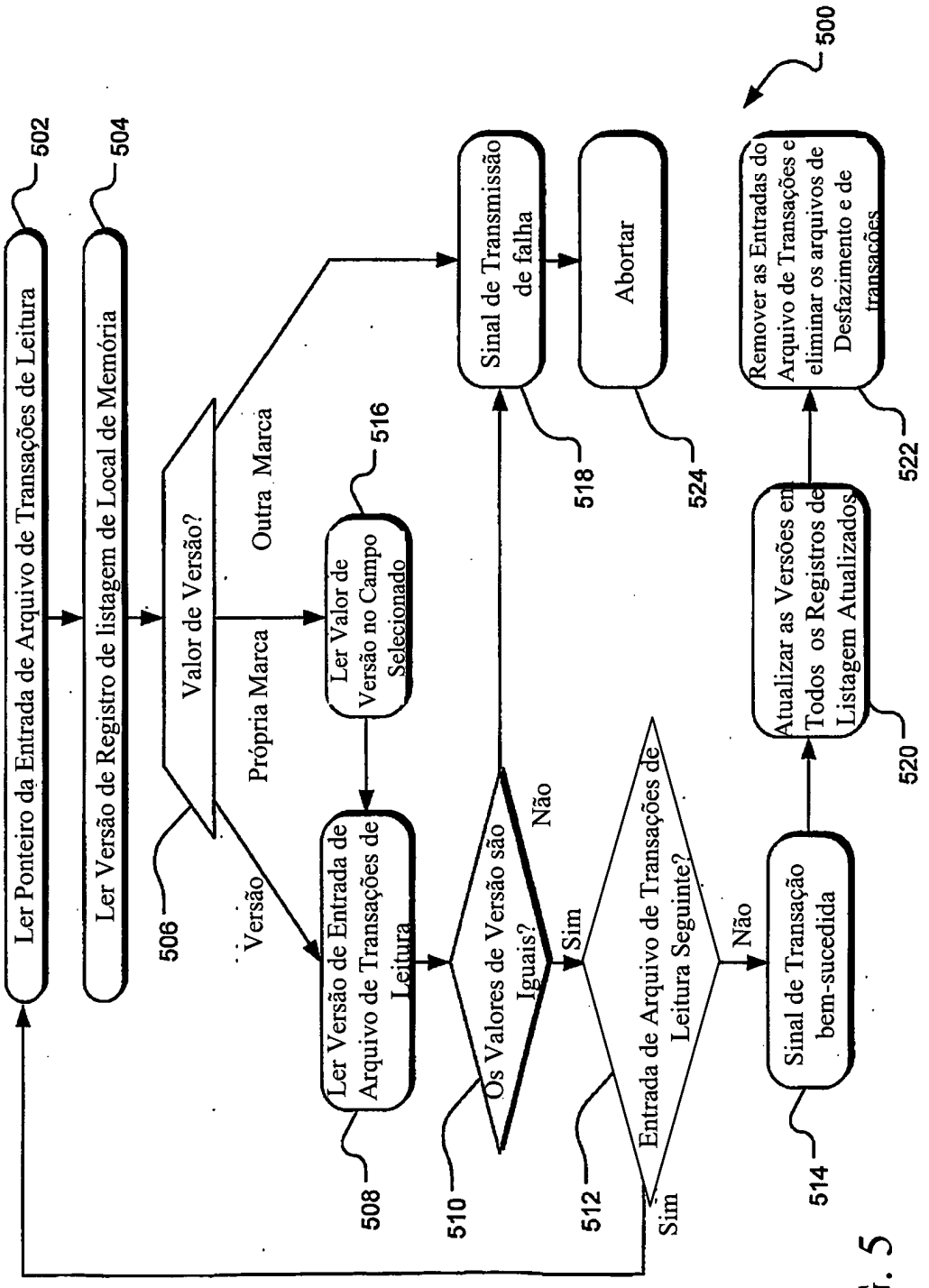
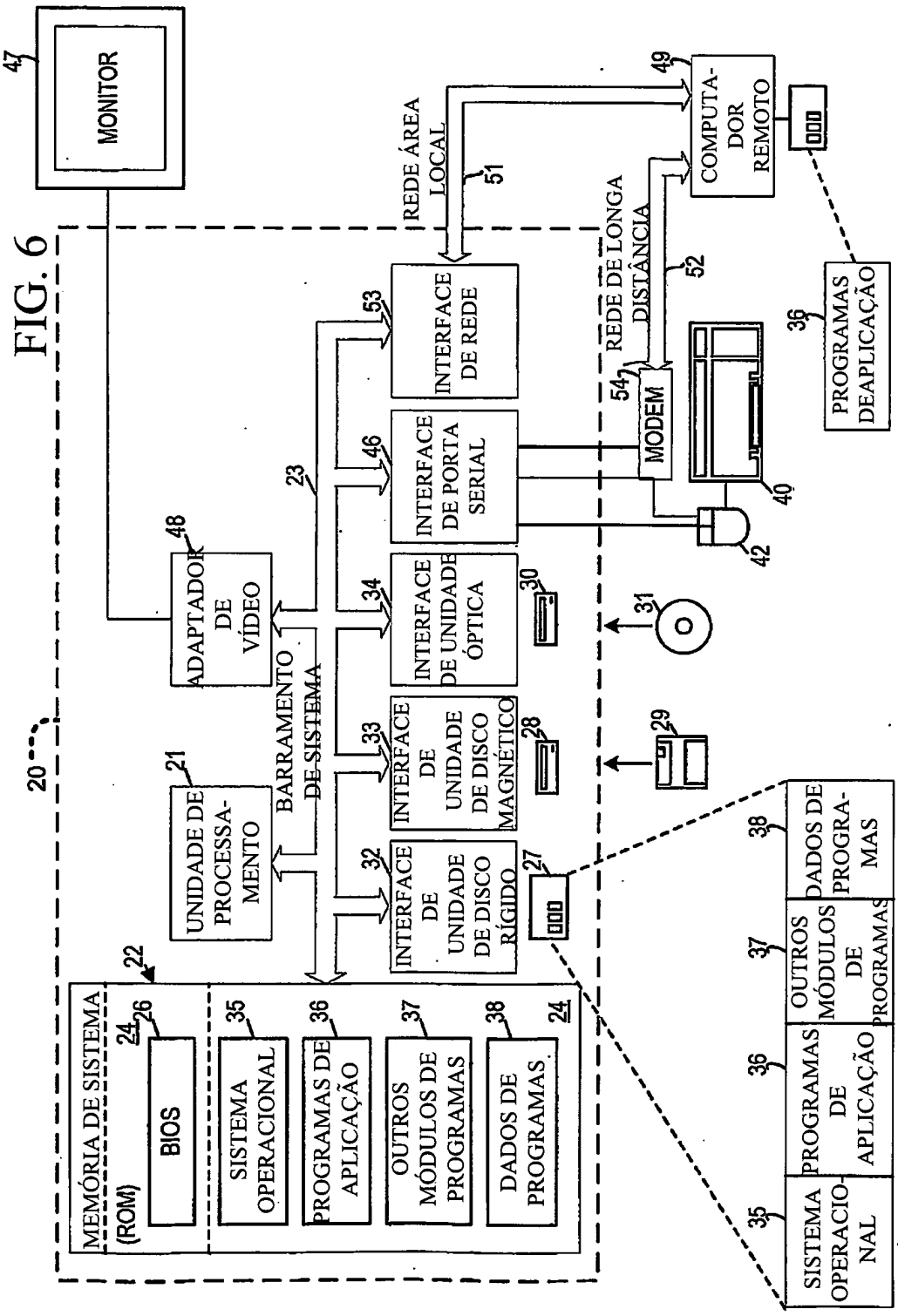

500
Ler Ponteiro da Entrada de Arquivo de Transações de Leitura
502
Ler Versão de Registro de listagem de Local de Memória
504
Valor de Versão?
506
Versão
Própria Marca
Outra Marca
Ler Valor de Versão no Campo Selecionado
516
Ler Versão de Entrada de Arquivo de Transações de Leitura
508
Os Valores de Versão são Iguais?
510
Não
Sim
Sinal de Transmissão de falha
518
Abortar
524
Entrada de Arquivo de Transações de Leitura Seguinte?
512
Sim
Não
Sinal de Transação bem-sucedida
514
Atualizar as Versões em Todos os Registros de Listagem Atualizados
520
Remover as Entradas do Arquivo de Transações e eliminar os arquivos de Desfazimento e de transações
522

FIG. 5

FIG. 6


MONITOR
47
REDE ÁREA LOCAL
51
COMPUTADOR REMOTO
49
REDE DE LONGA DISTÂNCIA
52
INTERFACE DE REDE
53
INTERFACE DE PORTA SERIAL
46
MODEM
54
PROGRAMAS DEAPLICAÇÃO
36
40
42
ADAPTADOR DE VÍDEO
48
23
INTERFACE DE UNIDADE ÓPTICA
34
30
31
INTERFACE DE UNIDADE DE DISCO MAGNÉTICO
33
28
29
INTERFACE DE UNIDADE DE DISCO RÍGIDO
32
27
BARRAMENTO DE SISTEMA
UNIDADE DE PROCESSA-MENTO
21
20
22
MEMÓRIA DE SISTEMA (ROM)
24
26
BIOS
SISTEMA OPERACIONAL
35
PROGRAMAS DE APLICAÇÃO
36
OUTROS MÓDULOS DE PROGRAMAS
37
DADOS DE PROGRAMAS
38
24
DADOS DE PROGRA-MAS
38
OUTROS MÓDULOS DE PROGRAMAS
37
PROGRAMAS DE APLICAÇÃO
36
SISTEMA OPERACIO-NAL
35

PI0613946-9

RESUMO

"MEMÓRIA TRANSACIONAL DE SOFTWARE DE ATUALIZAÇÃO DIRETA".

Uma interface de programação de memória transacional permite que uma corrente acesse, diretamente e com segurança, um ou mais locais de memória partilhada dentro de uma transação, enquanto mantendo estruturas de controle para gerenciar acessos de memória àqueles mesmos locais por uma ou mais correntes concorrentes. Cada local de memória acessado pela corrente é associado com um registro de listagem, e cada corrente mantém um arquivo de transações dos seus acessos de memória. Dentro de uma transação, uma operação de leitura é conduzida diretamente no local de memória, e uma operação de escrita é tentada diretamente no local de memória, em oposição a algum armazenamento temporário. A corrente pode detectar inconsistências entre o registro de listagem de um local de memória e o arquivo de transações da corrente, para determinar se os acessos de memória dentro da transação não são confiáveis e se a transação precisa ser tentada de novo.