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

(21) **BR 10 2013 019517-0 A2**

(22) Data de Depósito: 31/07/2013

(43) Data da Publicação: 13/01/2015
(RPI 2297)

(54) Título: MÉTODO PARA DESENVOLVER UM SOFTWARE EM UM AMBIENTE DE COMPUTAÇÃO PARALELA

(51) Int.Cl.: G06F9/44

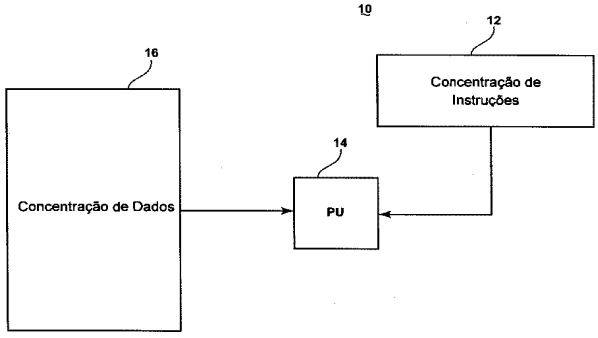
(52) CPC: G06F8/10; G06F8/20

(30) Prioridade Unionista: 17/08/2012 US 13/588,485

(73) Titular(es): GE AVIATION SYSTEMS LLC

(72) Inventor(es): BENJAMIN THOMAS OCCHIPINTI, ERIC DANIEL BUEHLER

(57) Resumo: MÉTODO PARA DESENVOLVER UM SOFTWARE EM UM AMBIENTE DE COMPUTAÇÃO PARALELA Trata-se de um método (100) para desenvolver o software em um ambiente de computação paralela que inclui, entre outras coisas, as etapas de desenvolvimento de uma implantação sequencial (114) e uma implantação paralela (118) do software e de verificação dos resultados da implantação paralela (120) do software contra os resultados da implantação sequencial (116) do software.



"MÉTODO PARA DESENVOLVER UM SOFTWARE EM UM AMBIENTE DE COMPUTAÇÃO PARALELA"

ANTECEDENTES DA INVENÇÃO

Processos de desenvolvimento de software têm sido desenvolvidos para melhorar a produtividade e a qualidade do produto de software durante todo ciclo de vida do programa. Já que os processadores de computação profissionais e pessoais têm migrado das unidades de processamento de núcleo único (CPU) para as CPUs de núcleo múltiplo e para as unidades de processamento gráfico (GPU), o software de computador está sofrendo uma transição de altamente sequencial para altamente paralelo.

A Figura 1 é um fluxograma esquemático de uma arquitetura de computador de processamento sequencial 10 com um fluxo de instruções único e fluxo de dados único. Processamento sequencial refere-se à execução dos códigos de computador em uma arquitetura de hardware em que um processador único executa um fluxo único de instruções a partir da concentração de instruções 12 para operar nos dados que são armazenados na memória na concentração de dados 16. A implantação sequencial do software que é programado pelo desenvolvedor de software reside na concentração de instruções 12. Um fluxo em série único das instruções é enviado a uma unidade de processamento (PU) única 14 a partir da concentração de instruções 12 à medida que o programa é executado. A PU 14 recebe dados da concentração de dados 16 em um fluxo em série único. A representação da arquitetura de processamento sequencial 10 é citada na literatura como um elemento da taxonomia de Flynn das arquiteturas de computadores, como instruções de fluxo único, dados de fluxo único (SISD). O exemplo mais conhecido dessa arquitetura são as máquinas de processador único tradicionais utilizadas nos PCs durante os anos 80 e 90.

A Figura 2 é um fluxograma esquemático de uma arquitetura de

computador de processamento paralelo 20 com um fluxo de instruções único e fluxo de dados múltiplo. O processamento paralelo refere-se a qualquer arquitetura de computador em que cálculos distintos podem ser executados simultaneamente. Em uma arquitetura de processamento paralelo, a execução de um código de computador paralelo ocorre onde os processadores múltiplos executam um fluxo único das instruções da concentração de instruções 22 para operar em fluxos múltiplos dos dados que são armazenados na memória na concentração de dados 26. A implantação paralela do software que é programada pelo desenvolvedor de software reside na concentração de instruções 22. Um fluxo em série único de instruções é enviado a um conjunto de unidades de processamento (PU) 24 a partir da concentração de instruções 22 à medida que o programa é executado. As PUs 24 recebem, cada uma, um fluxo de dados a partir da concentração de dados 26. Essa representação da arquitetura de processamento paralelo 20 é citada na literatura como um elemento da taxonomia de Flynn das arquiteturas de computadores como instruções de fluxo único, dados de fluxo múltiplo (SIMD). A arquitetura de SIMD é implantada nas GPUs e na maioria das CPUs de múltiplos núcleos.

Em um processo de desenvolvimento de software típico, os programadores escolhem um ambiente de processamento e o desenvolvem para o ambiente selecionado. Consequentemente, um desenvolvedor com a tarefa de criar um aplicativo que irá operar em uma plataforma de computação paralela que irá desenvolver uma implantação paralela de um produto de software em uma plataforma de computação paralela. Implantações paralelas de programas de computador são conhecidas por serem mais difíceis de gravar do que implantações sequenciais devido ao fato de que a concorrência dos cálculos adiciona o potencial para tipos adicionais de falhas de software e problemas de gerenciamento de recurso. Por exemplo, programas paralelos podem sofrer de um tipo de defeito conhecido como uma condição de corrida,

através da qual uma implantação paralela de um programa de computador exibe um comportamento anômalo devido a uma dependência crítica de temporização entre as operações executadas pelos processadores múltiplos. Esses tipos de erros podem ser extremamente difíceis de encontrar e, como resultado, pode ser muito demorado repará-los.

Processos de desenvolvimento de software têm sido desenvolvidos para melhorar a produtividade e a qualidade do produto de software durante todo ciclo de vida do programa. Tipicamente, essas metodologias são construídas mediante os modelos do ciclo de vida de desenvolvimento de software e fornecem um plano a ser seguido pelo desenvolvedor à medida que o programa de computador é desenvolvido e construído. Devido à natureza em série da arquitetura de hardware de computador que domina a indústria de computador pessoal há décadas, o processamento sequencial moldou os processos de desenvolvimento de software que os cientistas de computador têm utilizado para desenvolver o código de computador. O advento e a acessibilidade de unidades de processamento de gráfico (GPUs) e unidades de processamento central de múltiplos núcleos (CPUs) levaram a computação paralela ao uso corrente tanto da computação pessoal quanto empresarial. Como resultado, os processos de desenvolvimento de software são inadequados e ineficientes para o desenvolvimento e a distribuição do software paralelo.

A Figura 3 é um fluxograma esquemático do processo de projeto sequencial para o desenvolvimento de software conhecido como modelo em Cascata 30. O modelo em Cascata 30 é um processo de projeto bem conhecido que foi adaptado a partir da metodologia de projeto de hardware e usado em desenvolvimento de software, no qual cada fase do processo é concluída antes que a próxima fase se inicie. As fases do modelo em Cascata 30 são os requisitos 32, projeto 34, implantação 36, verificação 38 e

manutenção 40. Cada etapa é ligada sucessivamente à próxima etapa de modo que o processo de desenvolvimento global avance, após a conclusão da etapa atual, para a próxima etapa e é, muitas vezes, visualizado com as etapas anteriores acima das etapas seguintes de modo que a progressão seja análoga a uma cascata.

A etapa inicial do modelo em Cascata 30 para o desenvolvimento de software é o estágio de requisitos 32. O estágio de requisitos 32 é ligado sucessivamente ao estágio de projeto 34; assim, o estágio de requisitos 32 precisa estar completo antes que o estágio de projeto 34 possa começar. O estágio de projeto 34 é ligado sucessivamente ao estágio de implantação 36. O estágio de implantação 36 é ligado sucessivamente ao estágio de verificação 38. O estágio de verificação 38 é ligado sucessivamente ao estágio de manutenção 40. Após a conclusão do estágio de manutenção 40, o processo de desenvolvimento é completo.

Tipicamente o objetivo do estágio de requisitos 32 é descrever a finalidade do software e desenvolver uma especificação de requisitos de software. Uma especificação de requisitos de software é uma descrição completa de um sistema de software que define tanto requisitos funcionais quanto não funcionais do software a serem desenvolvidos. Os requisitos funcionais para o software são o conjunto das entradas, comportamentos e saídas que descrevem como o software funcionará. Os requisitos funcionais são tipicamente documentados no estágio de requisitos 32 como os cálculos e casos de uso. Os requisitos não funcionais para o software descrevem qualidades que o software irá exibir, como velocidade, estabilidade, capacidade e portabilidade. Os requisitos não funcionais são os critérios que irão ditar a arquitetura do sistema.

O estágio de projeto 34 é um processo de planejamento de uma solução de software específica que irá satisfazer o propósito e os requisitos

especificados anteriormente no estágio de requisitos 32. Durante o estágio de projeto 34, os desenvolvedores de software irão examinar os requisitos funcionais e não funcionais e desenvolver um modelo de software que irá detalhar o projeto fundamental do software a ser implantado. Considerações típicas durante o projeto de software são compatibilidade, modularidade, confiabilidade, usabilidade, robustez e etc. O projeto fundamental do modelo de software irá descrever uma hierarquia ou estrutura que descreve a arquitetura de software. A arquitetura de software irá descrever os módulos e componentes de software individuais e como os módulos irão se interconectar. A saída do estágio de projeto 34 é a documentação do modelo de software e pode ser uma descrição de texto simples, um fluxograma ou uma descrição hierárquica em uma linguagem de modelagem, como a Linguagem de Modelagem Unificada (UML).

O estágio de implantação 36 é a fase no ciclo de desenvolvimento de software em que o código de computador está, de fato, gravado. A descrição técnica do estágio de projeto é realizada como um programa ou componente de software. O programa é destinado a cumprir com os requisitos de software a partir do estágio de requisitos 32 por ser uma implantação direta do projeto de software do estágio de projeto 34.

O estágio de verificação 38 do ciclo de desenvolvimento de software é o processo em que o software implantado é testado quanto ao projeto e aos requisitos de software para comprovar que o programa foi construído corretamente e para a especificação. A verificação de software é um processo metódico em que os testes são gravados para validar que o software é executado conforme especificado. Se o software não passa nos testes de validação, o programa é depurado para encontrar e reduzir os defeitos. Após a conclusão do estágio de verificação 38 do processo de desenvolvimento em Cascata 30, o software é instalado e é feita a manutenção do mesmo. O

estágio de manutenção 40 do processo de desenvolvimento de software ocorre após o sistema de software ter sido instalado na plataforma de usuário final. Nesse momento, o usuário final irá identificar anteriormente falhas desconhecidas ou problemas de desempenho. Tipicamente, os sistemas de software desenvolvidos sob o modelo em Cascata 30 começaram a evoluir como requisitos de software de usuários finais que mudam com o uso de um sistema em campo. Portanto, o estágio de manutenção 40 é a fase em que um sistema de software se distancia dos requisitos de software e do projeto desenvolvido sob o modelo em Cascata 30 em resposta às necessidades dinâmicas do usuário final.

BREVE DESCRIÇÃO DA INVENÇÃO

A invenção refere-se a um método para desenvolver um software em um ambiente de computação paralela. O método compreende as etapas de desenvolver uma implantação sequencial do software em um ambiente de processamento sequencial; verificar se a implantação sequencial do software funciona conforme esperado; desenvolver uma implantação paralela da implantação sequencial do software em um ambiente de processamento paralelo e verificar os resultados da implantação paralela do software contra os resultados da implantação sequencial do software.

BREVE DESCRIÇÃO DOS DESENHOS

Nos desenhos:

A Figura 1 é um fluxograma esquemático de uma técnica anterior da arquitetura de computador de processamento sequencial com um fluxo de instruções único e um fluxo de dados único.

A Figura 2 é um fluxograma esquemático de uma técnica anterior da arquitetura de processamento paralelo com um fluxo de instruções único e um fluxo de dados múltiplos.

A Figura 3 é um fluxograma esquemático de uma técnica anterior

do processo de projeto sequencial para o desenvolvimento de software conhecido como modelo em Cascata.

A Figura 4 é um fluxograma esquemático do processo de projeto paralelo para o desenvolvimento simultâneo do software paralelo e sequencial, de acordo com uma realização da invenção.

A Figura 5 é um fluxograma esquemático do processo utilizado para verificar a implantação de código paralelo contra a implantação de código sequencial de acordo com uma realização da invenção.

DESCRIÇÃO DAS REALIZAÇÕES DA INVENÇÃO

Nos antecedentes e na descrição a seguir, com fins exemplificativos, diversos detalhes específicos são fornecidos a fim de fornecer uma compreensão completa da tecnologia descrita no presente documento. Será evidente para um versado na técnica, no entanto, que as realizações exemplificativas podem ser praticadas sem esses detalhes específicos. Em outros casos, estruturas e um dispositivo são mostrados em forma de diagrama a fim de facilitar a descrição das realizações exemplificativas.

As realizações exemplificativas são descritas com referência aos desenhos. Esses desenhos ilustram certos detalhes das realizações específicas que implantam um módulo, método ou produto de programa de computador descrito no presente documento. No entanto, os desenhos não devem ser interpretados como impondo quaisquer limitações que podem estar presentes nos desenhos. O produto de programa de computador e o método podem ser fornecidos em quaisquer meios legíveis por máquina para a realização de suas operações. As realizações podem ser programadas com o uso de um processador de computador existente ou por um processador de computador de finalidade especial incorporado a essa ou outra finalidade ou por um sistema conectado.

Conforme notado acima, as realizações descritas no presente

documento podem incluir um produto de programa de computador que compreende meios legíveis por máquina para portar ou ter estruturas de dados ou instruções executáveis por máquina armazenadas na mesma. Tais meios legíveis por máquina podem ser quaisquer meios disponíveis, os quais podem ser acessados por um computador de finalidade geral ou finalidade especial ou outra máquina com um processador. Com fins exemplificativos, tais meios legíveis por máquina podem compreender RAM, ROM, EPROM, EEPROM, CD-ROM ou outro armazenamento de disco óptico, armazenamento de disco magnético ou outros dispositivos de armazenamento magnético ou qualquer outro meio que possa ser utilizado para portar ou armazenar o código de programa desejado na forma de estruturas de dados ou instruções executáveis por máquina e que podem ser acessados por um computador de finalidade geral ou finalidade especial ou outra máquina com um processador. Quando as informações são transferidas ou fornecidas através de uma rede ou outra conexão de comunicação (diretamente conectado, sem fio ou uma combinação de diretamente conectado ou sem fio) para uma máquina, a máquina considera a conexão corretamente como um meio legível por máquina. Assim, qualquer conexão desse tipo é corretamente chamada de meio legível por máquina. As combinações dos anteriores também estão incluídas dentro do âmbito dos meios legíveis por máquina. As instruções executáveis por máquina compreendem, por exemplo, instruções e dados, os quais fazem com que um computador de finalidade geral, um computador de finalidade especial ou máquinas de processamento de finalidade especial executem uma certa função ou um grupo de funções.

As realizações serão descritas no contexto geral das etapas do método que podem ser implantadas em uma realização por um produto de programa que inclui instruções executáveis por máquina, como código de programa, por exemplo, na forma dos módulos de programa executados por

máquinas nos ambientes de rede. Em geral, os módulos de programa incluem rotinas, programas, objetos, componentes, estruturas de dados e etc. que tem o efeito técnico de executar tarefas particulares ou implantar tipos de dados abstratos particulares. As instruções executáveis por máquina, estruturas de dados associados e módulos de programa representam exemplos de código de programa para a execução de etapas do método exibido no presente documento. A sequência particular de tais instruções executáveis ou estruturas de dados associadas representam exemplos de atos correspondentes para implantar as funções descritas em tais etapas.

As realizações podem ser praticadas em um ambiente de rede que usa conexões lógicas a um ou mais computadores remotos que têm processadores. Conexões lógicas podem incluir uma rede de área local (LAN) e uma rede de área estendida (WAN) que estão presentes no presente documento com fins exemplificativos e não limitantes. Tais ambientes de rede são comuns em intranets, a internet e redes de computador estendidas no escritório ou estendidas na empresa e podem utilizar um grande variedade de protocolos de comunicação diferentes. Aqueles versados na técnica irão apreciar que tais ambientes de computação de rede irão abranger tipicamente muitos tipos de configuração de sistema de computador, incluindo computadores pessoais, dispositivos de mão, sistemas de multiprocessadores, eletrônicos de consumidor programável ou baseados em microprocessador, PCs de rede, minicomputadores, computadores de grande porte e similares.

As realizações também podem ser praticadas em ambientes de computação distribuídos, em que as tarefas são executadas por dispositivos de processamento remoto e local que estão unidos (também por enlaces conectados, enlaces sem fio ou por uma combinação de enlaces conectados ou sem fio) através de uma rede de comunicação. Em um ambiente de computação distribuído, os módulos de programa podem ser localizados em

dispositivos de armazenamento de memória tanto remota quanto local.

Um sistema exemplificativo para a implantação total ou porções das realizações exemplificativas pode incluir um dispositivo de computação de finalidade geral na forma de um computador, que inclui uma unidade de processamento, uma memória de sistema e um barramento de sistema, que acopla vários componentes de sistema que incluem a memória de sistema à unidade de processamento. A memória de sistema pode incluir memória somente de leitura (ROM) e memória de acesso randômico (RAM). O computador também pode incluir um acionador de disco rígido magnético para a leitura de um disco rígido magnético e para a gravação para o mesmo, um acionador de disco magnético para a leitura de um disco magnético removível ou gravação para o mesmo e um acionador de disco óptico para a leitura de um disco óptico removível ou gravação para o mesmo como um CD-ROM ou outros meios ópticos. Os acionadores e seus meios legíveis por máquina associados fornecem armazenamento não volátil das instruções executáveis por máquina, estruturas de dados, módulos de programa e outros dados para o computador.

Os efeitos técnicos do método exibido nas realizações incluem melhorar a eficiência do desenvolvimento do aplicativo de processamento paralelo. Do mesmo modo, o método melhora nas técnicas de depuração de software e hardware existentes habilitando as opções de depuração que não estão disponíveis de outro modo para um desenvolvedor de aplicativos em múltiplas plataformas de hardware, tanto sequencial quanto paralela. A implantação dessa técnica nas plataformas em campo irá melhorar a robustez do sistema através da seleção de modo automático do melhor estado de tempo de execução de um aplicativo.

A Figura 4 ilustra um modelo 100 de acordo com uma realização da invenção na qual a técnica anterior do modelo em Cascata 30 mostrada e

descrita em relação às Figuras 1 a 3 foram melhoradas de acordo com os exemplos ilustrativos fornecidos no presente documento. O modelo 100 é um processo de projeto de software para desenvolver o software em um ambiente de computação paralela. O processo de desenvolvimento aumenta o processo de projeto sequencial conhecido como modelo em Cascata 30 que é tipicamente utilizado no desenvolvimento de software de processamento sequencial. O aumento das fases demonstra o desenvolvimento simultâneo de uma implantação sequencial e paralela de um programa de software. E depois, o processo demostra um método para verificar a implantação paralela contra a implantação sequencial.

Os estágios do modelo em Cascata 100 modificado são os requisitos 110, o projeto 112, a implantação sequencial 114, a verificação sequencial 116, a implantação paralela 118, a implantação sequencial contra a implantação paralela 120 e a manutenção 122. O estágio de requisitos 110 é ligado sucessivamente ao estágio de projeto 112. O estágio de projeto 112 é ligado sucessivamente ao estágio de implantação sequencial 114. O estágio de implantação sequencial 114 é depois ligado a tanto verificação sequencial 116 quanto aos estágios de implantações paralelas 118. Tanto a verificação sequencial 116 quanto os estágios de implantações paralelas 118 são ligados ao estágio de implantação sequencial contra a implantação paralela 120. O estágio de implantação sequencial contra a implantação paralela 120 é ligado sucessivamente ao estágio de manutenção 122.

O estágio de requisitos 110 do modelo em Cascata 100 modificado é implantado como no estágio de requisitos 32 do modelo em Cascata 30. Adicionalmente aos processos do estágio de projeto 34 do modelo em Cascata 30, o estágio de projeto 112 do modelo em Cascata 100 modificado inclui a análise superficial adicional pelo desenvolvedor para determinar quais, se houver, algoritmos sequenciais serão convertidos para

algoritmos paralelos para a implantação paralela. A análise para determinar quais, se houver, algoritmos sequenciais serão convertidos para algoritmos paralelos para a implantação paralela ser limitada no estágio de projeto 112 devido aos requisitos iniciais do estágio de requisitos 110 e hipóteses e requisitos do estágio de projeto 112 podem mudar como os algoritmos são implantados no código de computador.

Após a conclusão do estágio de projeto 112, o estágio de implantação sequencial 114 é iniciado. O estágio de implantação sequencial 114, similar ao estágio de implantação 36 do modelo em Cascata 30, é a fase no ciclo de desenvolvimento de software em que o código de computador está gravado realmente. O código de computador desenvolvido no estágio de implantação sequencial 114 é uma implantação sequencial do software que executa em um ambiente de processamento sequencial. Os requisitos não funcionais desenvolvidos no estágio de requisitos 32 como a velocidade não necessitarão ser encontrados no estágio de implantação sequencial 114. Em uma realização alternativa da invenção, os algoritmos paralelos e sequenciais têm cada um, um conjunto de requisitos e a implantação sequencial precisam satisfazer os requisitos notados para uma implantação sequencial. Após a conclusão do estágio de implantação sequencial 114, uma análise mais completa determina quais algoritmos sequenciais se beneficiarão mais da paralelização.

O estágio de verificação sequencial 116 começa após a conclusão do estágio de implantação sequencial 114. Como o estágio de verificação 38 do modelo em Cascata 30, o objetivo do estágio de verificação sequencial 116 é para verificar se a implantação sequencial do software funciona conforme esperado.

Simultaneamente ao estágio de verificação sequencial 116, o estágio de implantação paralela 118 pode começar. O objetivo do estágio de

implantação paralela 118 é para desenvolver uma implantação paralela da implantação sequencial do software em um ambiente de processamento paralelo. Os ambientes de processamento paralelos podem incluir as arquiteturas de hardware de uso corrente como CPUs de múltiplos núcleos e GPUs. Até porque mesmo pequenas mudanças na implantação algorítmica podem resultar em mudanças substanciais a uma implantação paralela, o estágio de implantação de processamento paralelo 118 precisaria não ser iniciado até o estágio de implantação sequencial 114 ser concluído.

Após a conclusão tanto do estágio de verificação sequencial 116 quanto do estágio de implantação paralela 118, o estágio de verificação paralela 120 pode começar. O estágio de verificação paralela 120 verifica os resultados da implantação paralela do software contra os resultados da implantação sequencial do software.

Após a implantação paralela de o software ter sido verificada no estágio de verificação paralelo 120, o processo de desenvolvimento de software do modelo em Cascata 100 modificado entra o estágio de manutenção 122. Como o estágio de manutenção 40 do modelo em Cascata 30, o estágio de manutenção 122 do modelo em Cascata modificado 122 é a migração do plano de software do processo de desenvolvimento de software a um desenvolvimento evolutivo pelo qual os defeitos são revelados pelo usuário final e as mudanças incrementais ao software ocorrem devido àqueles defeitos e mudanças nos requisitos do usuário final.

A Figura 5 ilustra um processo 200 que pode ser utilizado para executar o estágio de verificação do código paralelo 120 do modelo em Cascata 100 modificado. Nesse processo 200, para verificar a implantação de código paralelo, uma sucessão de testes pode ser desenvolvida e executada pelo desenvolvedor de software para comparar a saída e os resultados da implantação sequencial verificada anteriormente do software. O processo pode

começar com o teste de segurança 210 da implantação paralela. Os resultados do teste de segurança podem ser comparados depois aos resultados da implantação sequencial 218. Sob a comparação bem sucedida dos resultados, o processo pode continuar com teste de unidade 212 da implantação paralela. Os resultados do teste de unidade podem ser comparados depois aos resultados da implantação sequencial 218. Sob a comparação bem sucedida dos resultados, o processo pode continuar com o teste do tipo corner 214 da implantação paralela. Os resultados do teste do tipo corner podem ser comparados depois aos resultados da implantação sequencial 218. Sob a comparação bem sucedida dos resultados, o processo pode continuar com o teste do tipo edge 214 da implantação paralela. Os resultados do teste do tipo edge podem ser comparados depois aos resultados da implantação sequencial 218.

Mediante o início da verificação de implantação paralela, o desenvolvedor de software pode desenvolver e executar o teste de segurança 210. O teste de segurança é tipicamente um método informal de teste destinado a verificar rapidamente o funcionamento total de uma implantação de software. Devido a uma sucessão de teste que já podem ter sido desenvolvidos para a implantação sequencial no estágio de verificação sequencial 116, os resultados do teste de segurança 210 podem ser comparados aos resultados para os testes similares desenvolvidos para o estágio de verificação sequencial 116.

Após os resultados do teste de segurança terem sido verificados contra os resultados da implantação sequencial, o desenvolvedor de software pode desenvolver e executar o teste de unidade 212. O teste de unidade é um método de teste dos componentes de software que verificam a funcionalidade de uma seção específica do código de software. Devido a uma sucessão de testes de unidade que podem já ter sido desenvolvidas para a implantação

sequencial no estágio de verificação sequencial 116, os resultados do teste de unidade 212 podem ser comparados aos resultados para os testes similares desenvolvidos para o estágio de verificação sequencial 116.

Após os resultados do teste de unidade terem sido verificados contra os resultados da implantação sequencial, o desenvolvedor de software pode desenvolver e executar o teste do tipo corner 214. Teste de canto é um método de teste dos casos patológicos em que o software precisa manipular as entradas que somente ocorrem fora dos parâmetros operacionais normais. Devido a uma sucessão de testes do tipo corner que podem já ter sido desenvolvidos para a implantação sequencial no estágio de verificação sequencial 116, os resultados do teste do tipo corner 214 podem ser comparados aos resultados para os testes similares desenvolvidos para o estágio de verificação sequencial 116.

Após os resultados do teste do tipo corner terem sido verificados contra os resultados da implantação sequencial, o desenvolvedor de software pode desenvolver e executar o teste do tipo edge 216. Teste do tipo edge é um método de teste dos casos patológicos em que o software precisa manipular as entradas em que somente um parâmetro único ocorre do fora dos parâmetros operacionais normais. Devido a uma sucessão de testes de borda que podem já ter sido desenvolvida para a implantação sequencial no estágio de verificação sequencial 116, os resultados do teste do tipo edge 216 podem ser comparados aos resultados para os testes similares desenvolvidos para o estágio de verificação sequencial 116.

Para verificar os resultados da implantação paralela do software contra os resultados da implantação sequencial do software, o desenvolvedor de software pode executar o teste de segurança nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial do estágio de verificação sequencial. Adicionalmente, o

desenvolvedor irá executar os testes de unidade, testes do tipo corner e testes do tipo edge nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial do estágio de verificação sequencial.

Com o uso do método cascata 100 modificado como uma ferramenta de desenvolvimento paralelo de software, o desenvolvedor de software pode desenvolver implantações paralelas com mais flexibilidade, permitindo que o desenvolvedor teste as implantações paralelas de modo confidencial e rápido. Como uma ferramenta de desenvolvimento de software, o método cascata 100 modificado de acordo com uma realização da presente invenção, permite a geração automática dos múltiplos cenários de teste paralelos distintos após a conclusão do estágio de verificação sequencial 116. Devido ao teste de segurança das implantações paralelas ser conhecido por ser difícil e demorado, o método cascata 100 modificado de acordo com uma realização da invenção fornece uma maneira rápido e segura de checar os resultados para uma implantação paralela através da comparação dos resultados checados previamente a partir do estágio de verificação sequencial 116.

Mediante do desenvolvimento de software com método cascata 100 modificado como uma ferramenta de desenvolvimento paralelo de software, teste de desempenho e depuração em campo das arquiteturas de hardware para o processamento paralelo é muito melhorada. Articulação do software desenvolvido entre as implantações sequenciais e paralelas permite a determinação de referência da arquitetura de hardware. A implantação sequencial fornece uma avaliação de linha de base da velocidade do hardware. Com o uso da referência de implantação sequencial, um programador pode fazer comparações informadas das referências da implantação paralela através de múltiplas arquiteturas de hardware diferentes. Por exemplo, um

programador ou usuário final pode trocar para diversos cartões de GPU diferentes em um sistema de hardware e coletar métricas de desempenho para comparação. Adicionalmente, o programador ou usuário final pode comparar as métricas de desempenho dos cartões GPU e um sistema de hardware integrado com um CPU.

Do mesmo modo, depuração em campo pode ser melhorada através do desenvolvimento de software com o método cascata 100 modificado devido ao fato de que o programador pode trocar a implantação sequencial do software para a implantação paralela se um erro ocorrer no campo. Por exemplo, um defeito de GPU pode ser isolado pela execução tanto do código paralelo quanto do código sequencial na GPU com endereçamento de memória separada escolhido para que o código sequencial isole cadeias de processamento na GPU. Articulação da implantação pode ser utilizada para isolar a falha como um defeito de hardware ou software. Por exemplo, um programador determina que um programa funciona com uma primeira configuração de hardware, mas não funciona com uma secunda configuração de hardware. Por estar apto a articular entre as duas configurações, o programador pode isolar mais rapidamente a função específica e localização alvo da falha no código.

Outro benefício do método cascata 100 modificado como uma ferramenta de desenvolvimento paralelo de software é para a distribuição do produto de software. Devido aos resultados de processo de desenvolvimento em tanto implantação sequencial quanto paralela do software, é obtido um aumento na robustez da plataforma. A ferramenta de desenvolvimento de software pode ser configurada para detectar se um estado de tempo de execução do software pode suportar uma implantação de processamento paralelo na plataforma para qual a mesma foi distribuída. Por exemplo, a ferramenta de desenvolvimento de software pode distribuir a implantação

sequencial em um sistema com CPU de núcleo único e a implantação paralela em um Sistema baseado em GPU.

Outro benefício do método Cascata 100 modificado é o alto nível de reutilização dos resultados dos estágios de requisitos e verificação antes das implantações paralelas e sequenciais. Por não precisar passar por esses estágios individualmente para cada implantação resultante, o programador pode economizar tempo e desenvolver mais rapidamente implantações funcionais. Similarmente, ter um conjunto de resultado de validação para comparar e verificar ambas as implantações é um benefício que economiza tempo.

Esta descrição escrita usa exemplos para revelar a invenção, incluindo o melhor modo, e também para permitir que qualquer pessoa versada na técnica pratique a invenção, o que inclui produzir e usar quaisquer dispositivos ou sistemas e executar quaisquer métodos incorporados. O âmbito patenteável da invenção é definido pelas reivindicações e pode incluir outros exemplos que ocorrem àqueles versados na técnica. Tais outros exemplos são destinados a estar dentro do âmbito das reivindicações se os mesmos têm elementos estruturais que não diferem da linguagem literal das reivindicações, ou se os mesmos incluem elementos estruturais equivalentes com diferenças insubstanciais das linguagens literais das reivindicações.

REIVINDICAÇÕES

1. MÉTODO PARA DESENVOLVER UM SOFTWARE EM UM AMBIENTE DE COMPUTAÇÃO PARALELA, que compreende as etapas de:

desenvolver uma implantação sequencial do software em um ambiente de processamento sequencial;

verificar se a implantação sequencial do software funciona conforme esperado;

desenvolver uma implantação paralela da implantação sequencial do software em um ambiente de processamento paralelo;

verificar os resultados da implantação paralela do software contra os resultados da implantação sequencial do software.

2. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, que compreende, ainda, a etapa de manter tanto a implantação sequencial do software quanto a implantação paralela do software.

3. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, que compreende, ainda, as etapas de:

determinar requisitos para o software; e

determinar um projeto para o software;

em que essas etapas são executadas anteriormente à etapa de desenvolvimento do software no ambiente de processamento sequencial.

4. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, em que a etapa de desenvolvimento de uma implantação paralela da implantação sequencial do software em um ambiente de processamento paralelo é executada, em geral, simultaneamente à etapa de verificação de que a implantação sequencial do software funciona conforme esperado.

5. MÉTODO PARA DESENVOLVER UM SOFTWARE, de

acordo com a reivindicação 1, em que o ambiente de processamento paralelo compreende pelo menos uma GPU.

6. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, em que o ambiente de processamento paralelo compreende pelo menos um processador de múltiplos núcleos.

7. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, em que o ambiente de processamento sequencial compreende pelo menos uma CPU.

8. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, em que a etapa de verificação dos resultados da implantação paralela do software contra os resultados da implantação sequencial do software compreende pelo menos uma das etapas a seguir:

(a) executar o teste de segurança nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial;

(b) executar pelo menos um teste de unidade nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial;

(c) executar pelo menos um teste do tipo corner nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial; e

(d) executar pelo menos um teste do tipo edge nos resultados da implantação paralela do software e comparar os resultados aos resultados da implantação sequencial.

9. MÉTODO PARA DESENVOLVER UM SOFTWARE, de acordo com a reivindicação 1, que compreende, ainda, a etapa de detectar se um estado de duração do software pode suportar o processamento paralelo.

10. MÉTODO PARA DESENVOLVER UM SOFTWARE, de

acordo com a reivindicação 9, compreende, ainda, a etapa de funcionamento do software no modo de processamento paralelo baseado em um sinal da etapa de detectar se um estado de duração do software pode suportar o processamento paralelo.

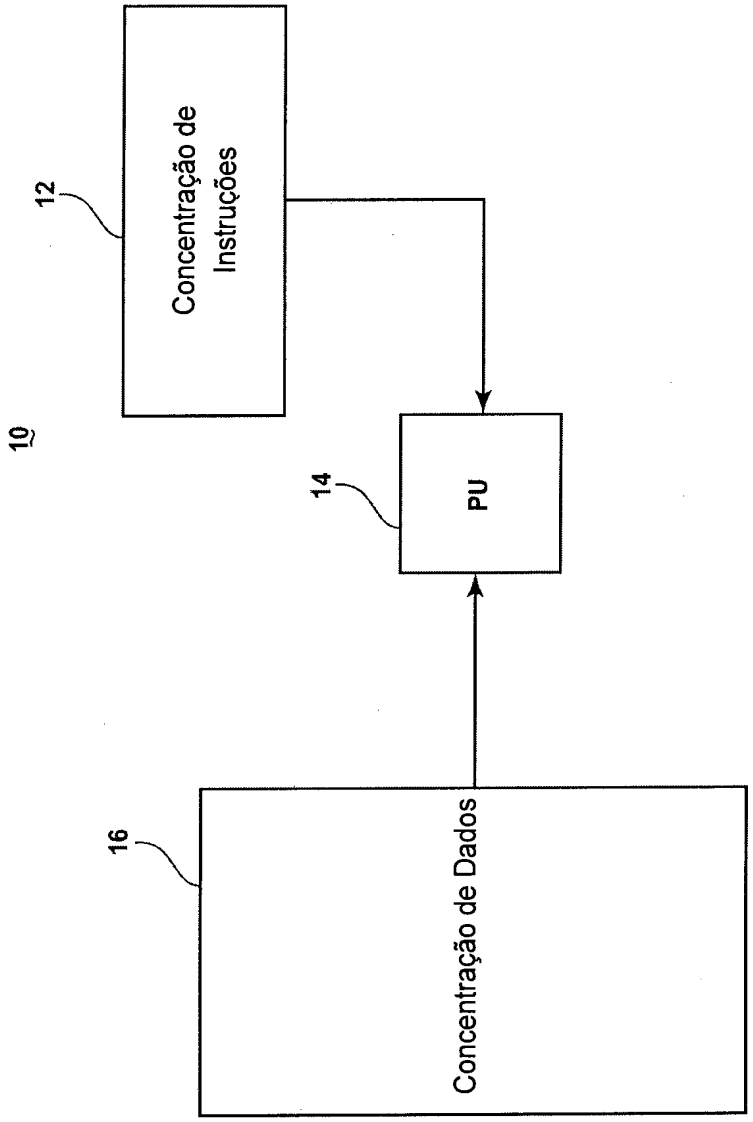


Fig. 1 (Técnica anterior)

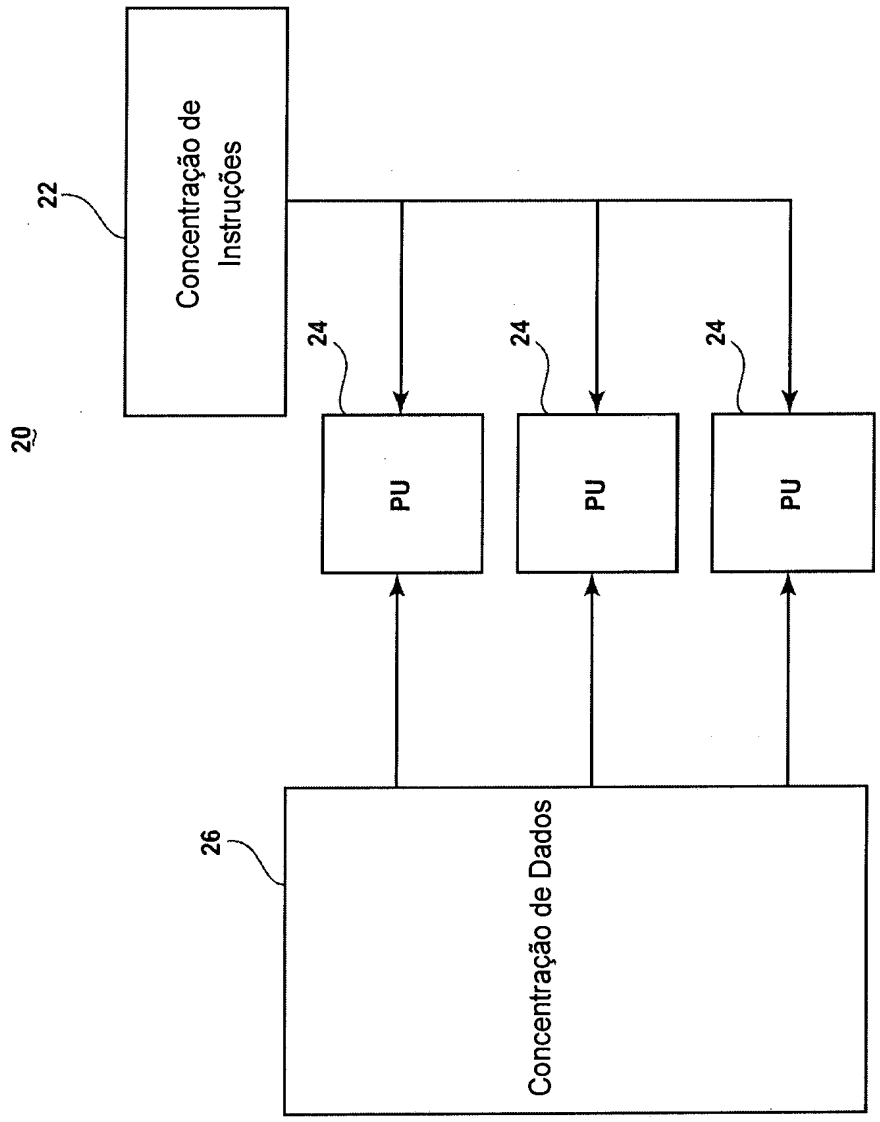


Fig. 2 (Técnica anterior)

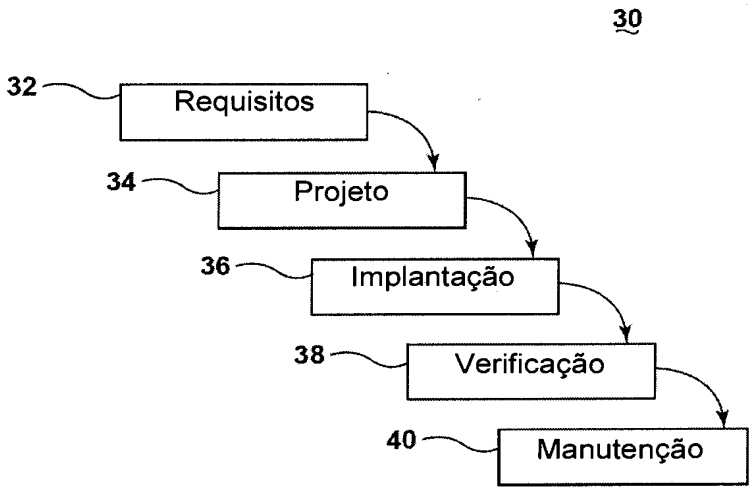


Fig. 3 (Técnica anterior)

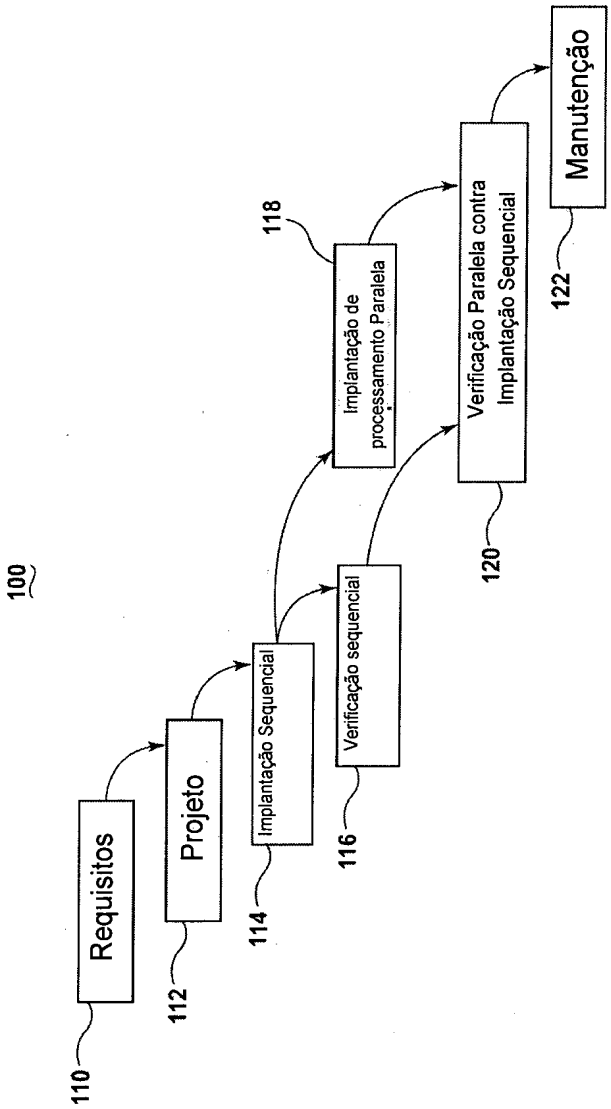

100
Requisitos
Projeto
Implantação Sequencial
Verificação sequencial
Implantação de processamento Paralela
Verificação Paralela contra Implantação Sequencial
Manutenção
110
112
114
116
118
120
122

Fig. 4

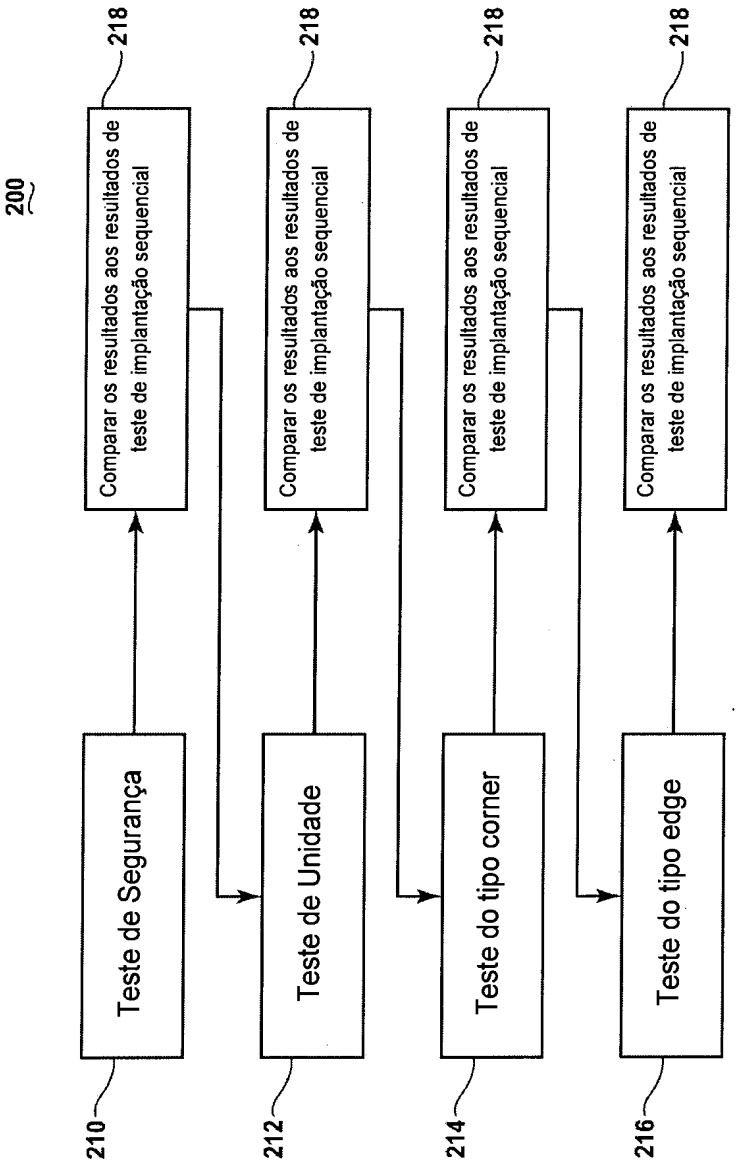

200
Teste de Segurança
Teste de Unidade
Teste do tipo corner
Teste do tipo edge
Comparar os resultados aos resultados de teste de implantação sequencial
Comparar os resultados aos resultados de teste de implantação sequencial
Comparar os resultados aos resultados de teste de implantação sequencial
Comparar os resultados aos resultados de teste de implantação sequencial
210
212
214
216
218
218
218
218

Fig. 5

RESUMO

"MÉTODO PARA DESENVOLVER UM SOFTWARE EM UM AMBIENTE DE COMPUTAÇÃO PARALELA"

Trata-se de um método (100) para desenvolver o software em um ambiente de computação paralela que inclui, entre outras coisas, as etapas de desenvolvimento de uma implantação sequencial (114) e uma implantação paralela (118) do software e de verificação dos resultados da implantação paralela (120) do software contra os resultados da implantação sequencial (116) do software.