

NIDays09

CONFERÊNCIA TECNOLÓGICA SOBRE
PROJETO GRÁFICO DE SISTEMAS





Programação Multicore com LabVIEW

Alisson Kokot – Engenheiro de Vendas
Osvaldo Santos – Engenheiro de Aplicações

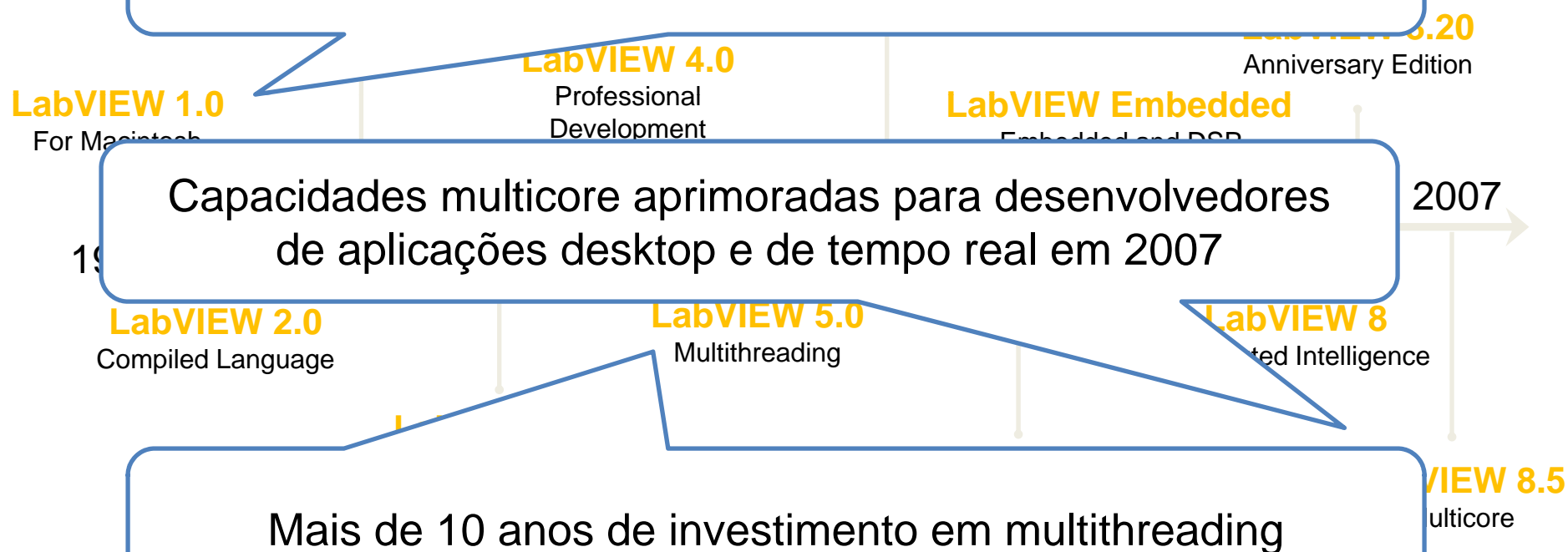


Introdução



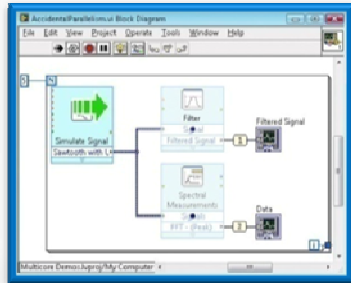
20 Anos de Inovação

Mais de 20 anos de investimento em programação em paralelo



Modelos de Projeto de Alto Nível

Fluxo de Dados

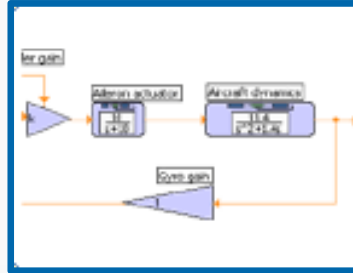


Linguagem C

```
#include "c:\devkit\include\math.h"
...
static void init(void)
{
    // Initialize variables
    // ...
}

// Main function
int main(void)
{
    // ...
    // Call the C function
    // ...
    return 0;
}
```

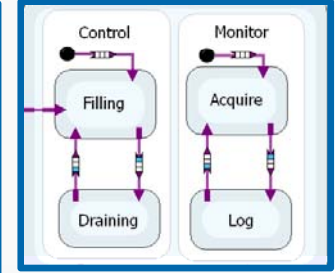
Simulação



Matemática Textual

```
1 c = 0.285 + 0.013i;
2 [X Y] = meshgrid(x, y);
3 z = X + i*Y;
4 for k=1:30
5     z = z.^2 + c;
6 end
```

Statechart



LabVIEW

Programação Gráfica

Linux



Macintosh

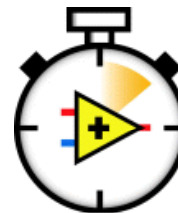


Windows

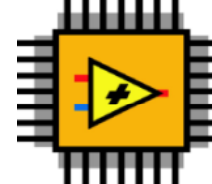


Plataformas Desktop

Real-Time



FPGA



MPU



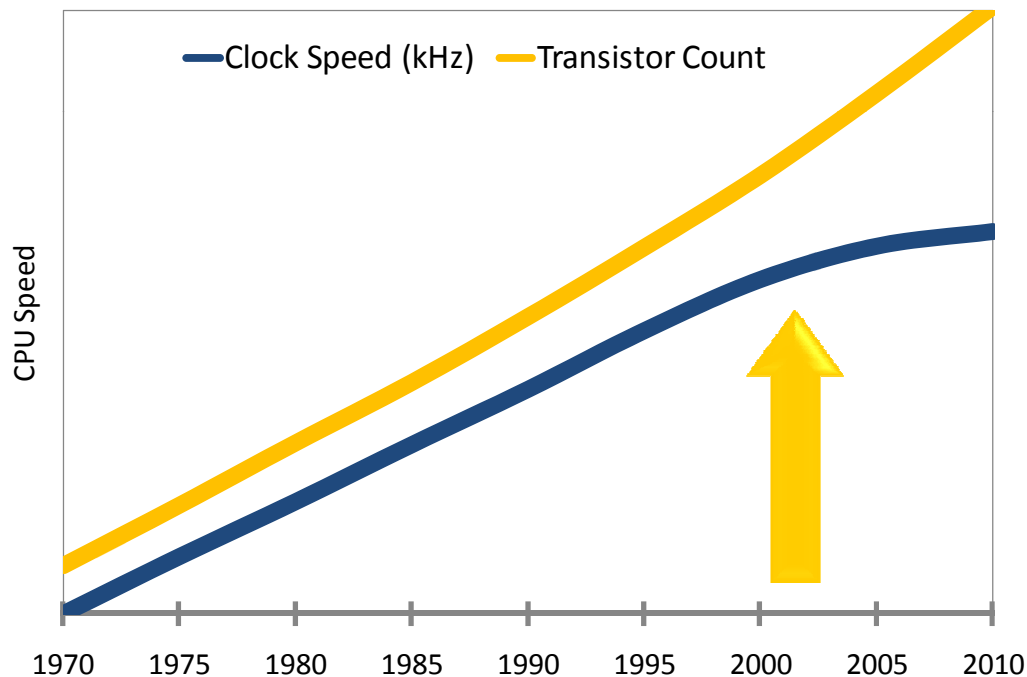
Plataformas Embarcadas

Desempenho da Arquitetura Paralela

Processadores
mais rápidos



Processadores
multicore

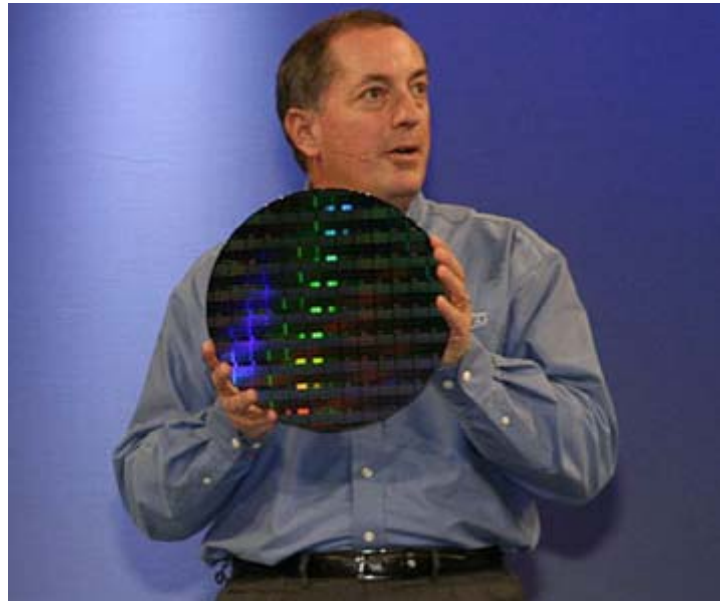


Processador Intel QX6700 Quad Core

- 4 processadores (par de pastilhas Core 2)
- Velocidade do clock de 2.66 GHz

“Intel promete 80 núcleos em cinco anos”

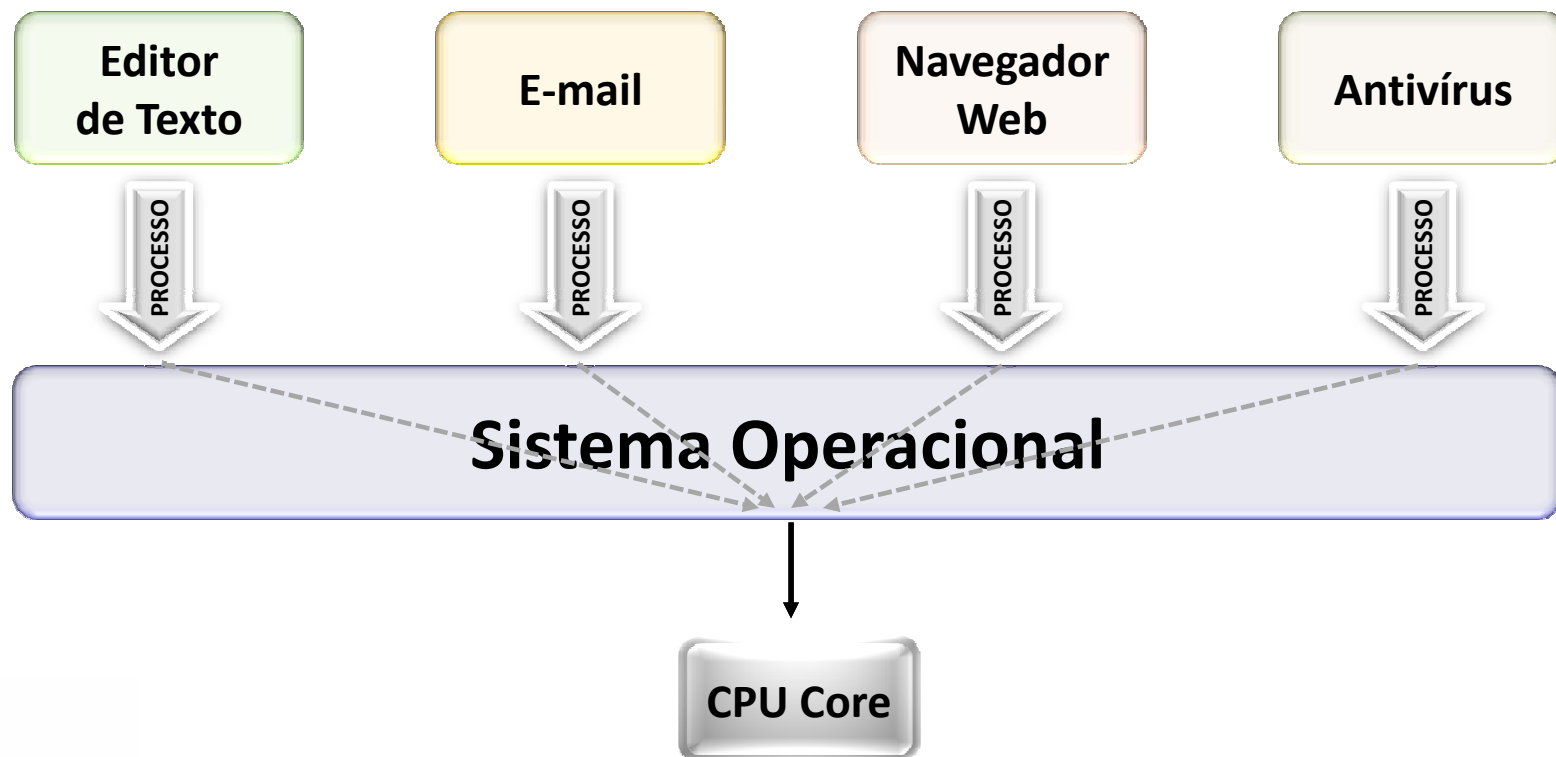
- Manchete do Forum de Desenvolvedores Intel
Setembro de 2006



Multicore >> Manycore

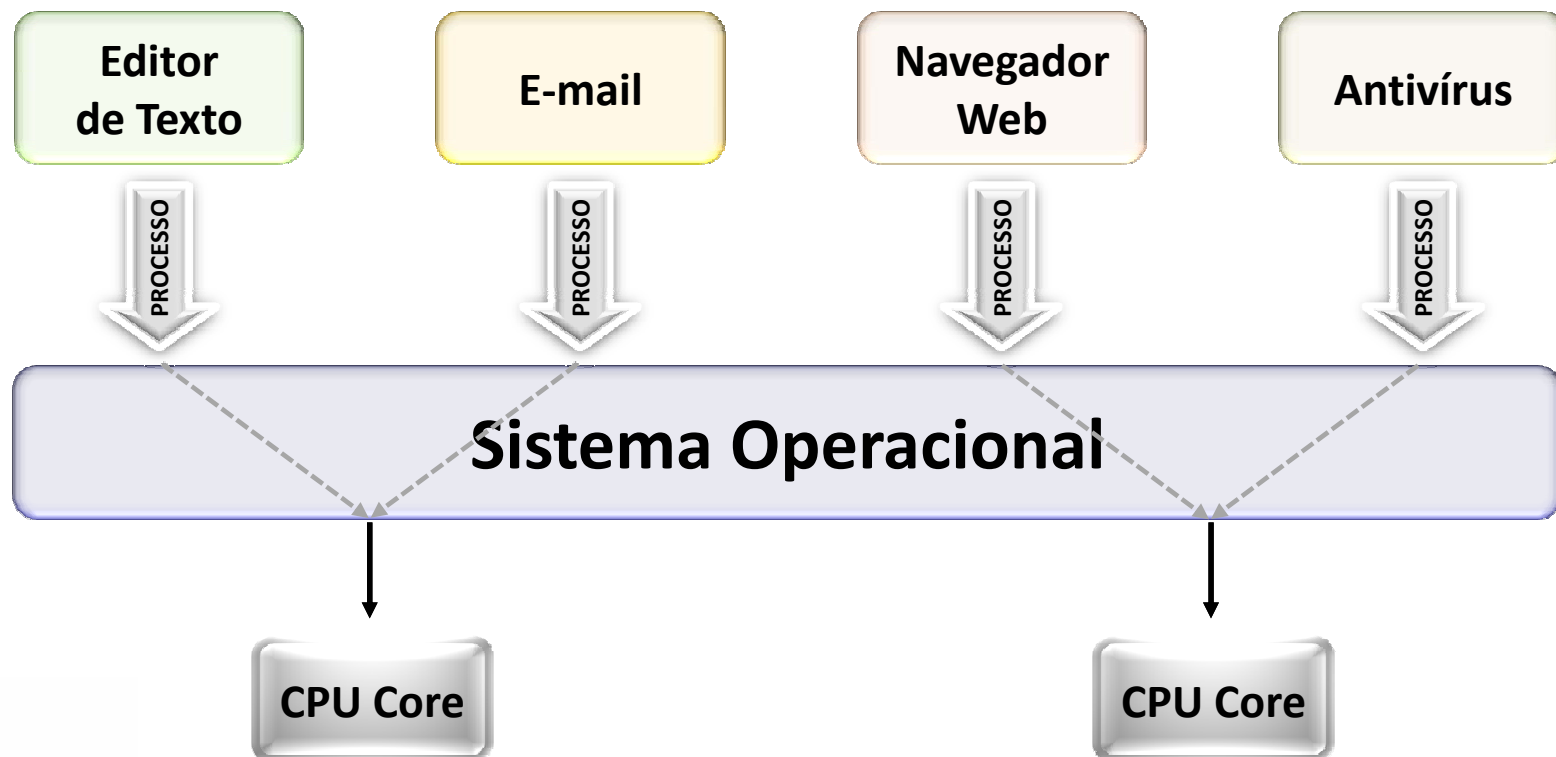
Multitasking

Sistemas operacionais agendam cada aplicação para compartilharem o tempo do processador.



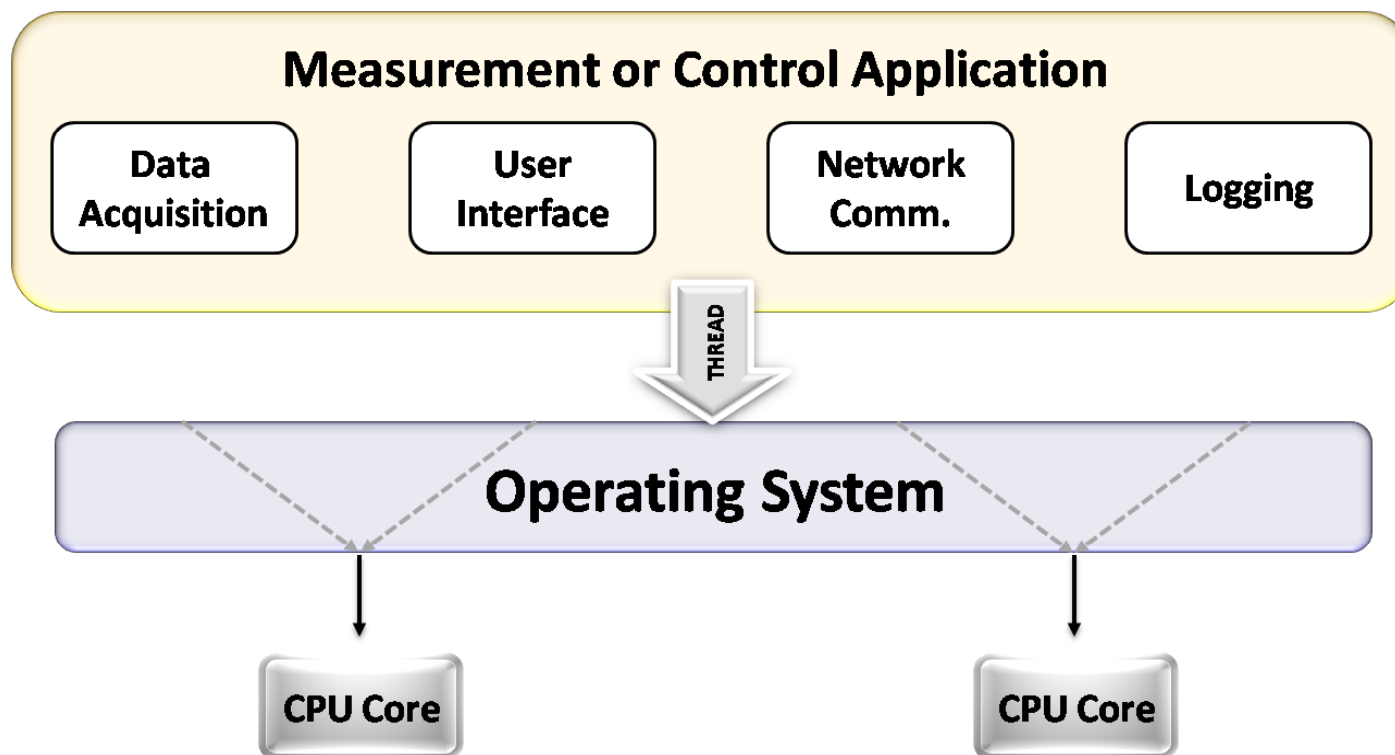
Multitasking com Múltiplos Núcleos

Sistemas operacionais agendam e automaticamente balanceiam as tarefas entre os múltiplos processadores.



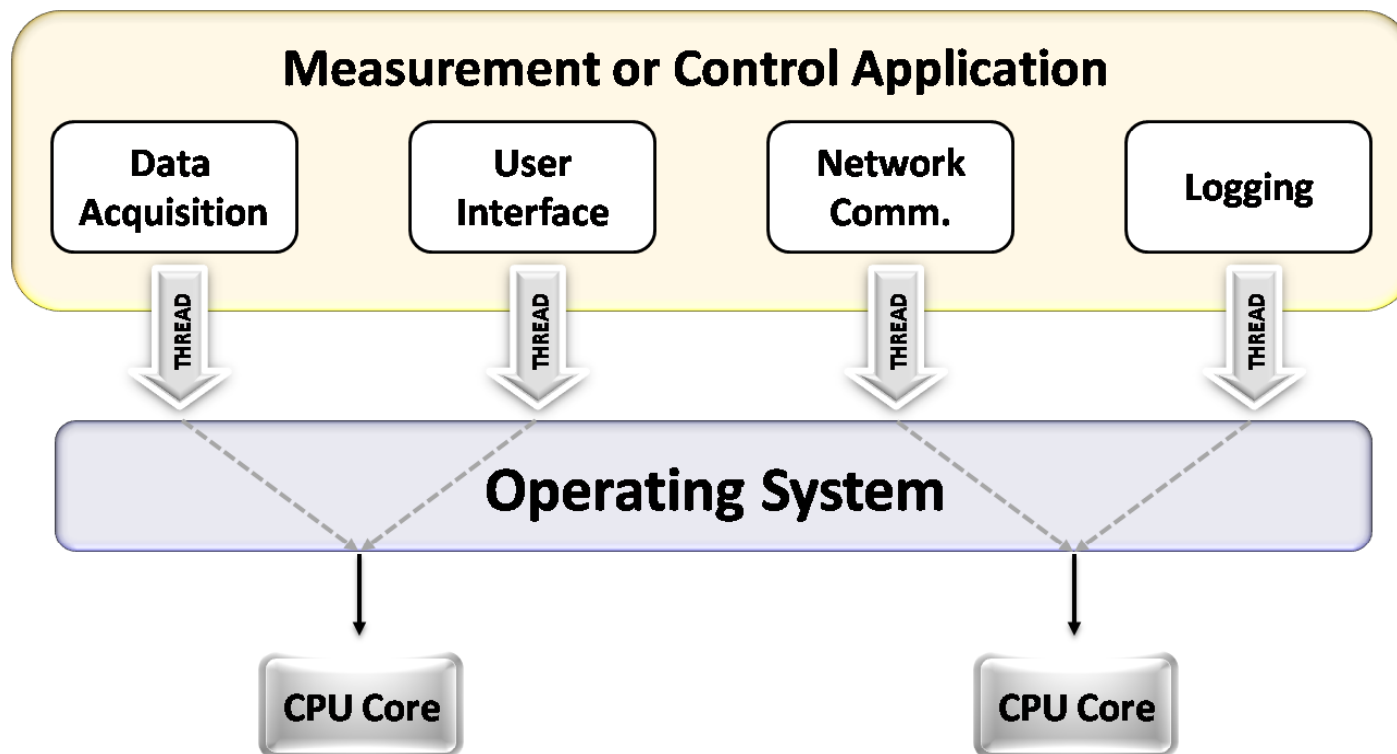
Impacto dos Multicore nos Desenvolvedores

Aplicações embarcadas estão tipicamente em sistemas dedicados (i.e. pouco multitasking).



Criando Aplicações com Multithread

Os desenvolvedores **devem** usar threads para se beneficiar dos processadores multicore.



O Desafio:

*“Você pode comprar laptops, desktops, celulares, PDAs, qualquer coisa multicore....Por causa dessa adoção generalizada, desenvolvedores de software tem finalmente se deparado com o fato de que **a programação como eles conheciam não será mais a mesma** - eles farão programação em paralelo daqui para frente”*

Dr. Anant Agarwal (MIT)

O Desafio:

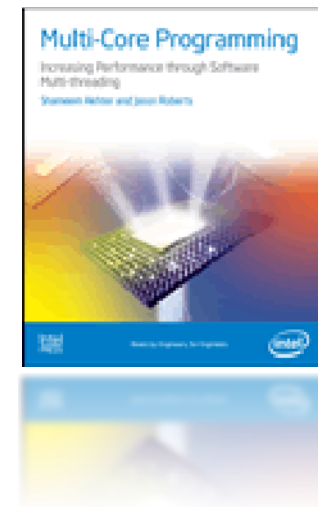
“Para explorar plenamente o poder dos processadores trabalhando em paralelo...os novos aplicativos de software devem lidar com o problema da concorrência.”

Bill Gates, Microsoft

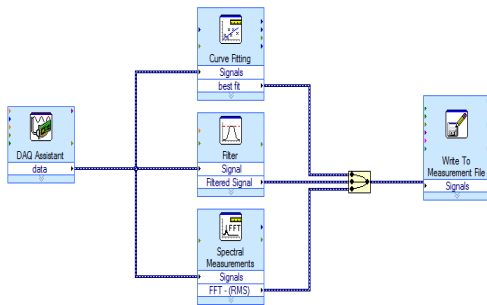
Programação com Threads é Difícil

Desenvolvedores usando ferramentas convencionais devem aprender novas funções e técnicas para:

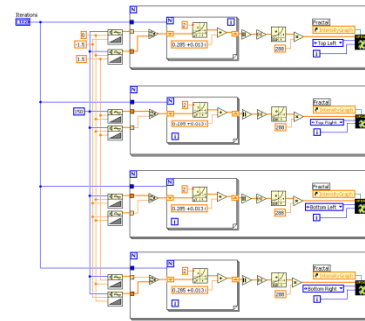
- Criar e destruir threads
- Fazer comunicação entre threads
- Sincronizar threads
- Depurar threads



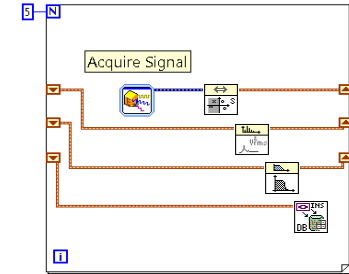
Programação em Paralelo com LabVIEW



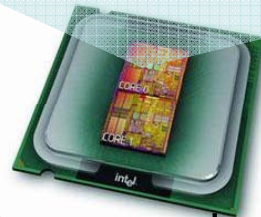
Paralelismo de Tarefas



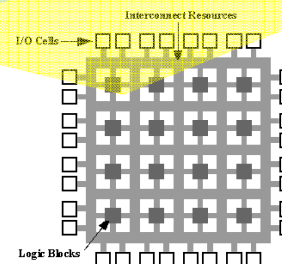
Paralelismo de Dados



Pipelining



Processadores
Multicore



FPGAs

Adotando Tecnologias Multicore : Considerações Chave e Benefícios de Negócio

Considerações Chave:

Pensar em Paralelo no Desenvolvimento de Aplicações

- Em 1998, a Honeywell-Measurex desenvolveu o “estado da arte” em máquinas de processamento de papel
- Utilizou toda capacidade de multithreading do LabVIEW 5.0 para substituir o código tradicional
- O sistema teve vantagem devido aos servidores com múltiplos processadores, mais de 1000 sistemas implantados em campo



Fonte:

1998 Measurement and Automation Newsletter em www.ni.com

Considerações Chave:

Definir uma Estratégia para Aprimorar o Código Tradicional em Aplicações Multithread

- A NASA criou uma estrutura LabVIEW para uma arquitetura multithread de tempo real
- Um algoritmo matemático específico criado em C foi importado no LabVIEW com o Call Library Node



“O benefício inicial de usar o LabVIEW Real-Time 8.5 e o suporte para SMP foi que, com menos carga na CPU, nós pudemos adicionar mais canais de aquisição de dados e matemática computacional à nossa aplicação de tempo real em LabVIEW.”

Kevin McDevitt, Advanced Concepts Group Leader
NASA Ames Research Center

Novas Aplicações Possíveis Devido ao Multicore:

Controle de Veículos Autônomos

- LabVIEW realiza o monitoramento do veículo em dois servidores HP de quatro núcleos no veículo Victor Tango o qual está classificado entre os 3 melhores no Urban Challenge.



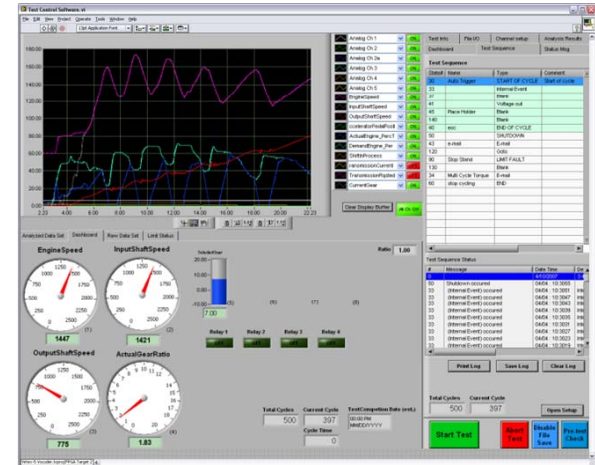
*“A capacidade do LabVIEW de automaticamente gerenciar o agendamento multithread para nossa aplicação **reduziu drasticamente** nosso tempo de desenvolvimento.”*

Michael Fleming
Presidente da Torc Technologies



Novas Aplicações Possíveis Devido ao Multicore: Sistema de Teste Portátil com Elevada Quantidade de Canais

- Eaton criou um sistema de teste portátil para uso embarcado em caminhões usando LabVIEW
- O sistema anterior realizava aquisição e análise de 16 canais em um único núcleo
- Agora são mais de 80 canais em multicore



*“Não houve **nenhuma necessidade de reescrever nossa aplicação** para a nova plataforma de processamento multicore.”*

Scott Sirine, Lead Design Engineer
Eaton Truck Division

EATON

Novas Aplicações Possíveis Devido ao Multicore: Processamento em Tempo Real de Alto Desempenho

- Controle do Tokamak no instituto Max Planck (Munich, Alemanha)
- LabVIEW Real-Time em um sistema de oito núcleos para controlar o plasma



*“...com LabVIEW, nós obtivemos um **aumento na velocidade de processamento de 20 vezes** em um processador de oito núcleos comparado a um processador de único núcleo...”*

Louis Giannone, Lead Project Researcher
Max Planck Institute



Max-Planck-Institut
für Plasmaphysik

Deferencial Competitivo:

Processamento mais Rápido para Aplicações Embarcadas e em Tempo Real

A Wineman Technology criou uma solução de testes em malha fechada baseada em LabVIEW Real-Time



*“Com LabVIEW Real-Time 8.5, nós **elevamos nossa taxa máxima de loop em 40%** ao utilizar ambos os núcleos do processador...”*

Roy Krans, Product Group Manager
Wineman Technology, Inc.



Diferencial Competitivo: Testes Mais Rápidos para Comunicações

A AmFax Ltd. (Reino Unido) criou um sistema de teste para wireless na próxima geração de telefones usando uma arquitetura de software pipeline em LabVIEW para processar algoritmos em paralelo



*“Com LabVIEW e uma controladora dualcore nós **atingimos aumento na velocidade de teste de até 5 vezes...**”*

Mark Jewell, Business Development Manager
Wireless, AmFax Ltd.



Conclusão

Considerações Chave

- Programação multicore exige que os desenvolvedores “pensem em paralelo”
- Para migração de código, uma estratégia deve ser estabelecida para transformar o código original para uma arquitetura paralela

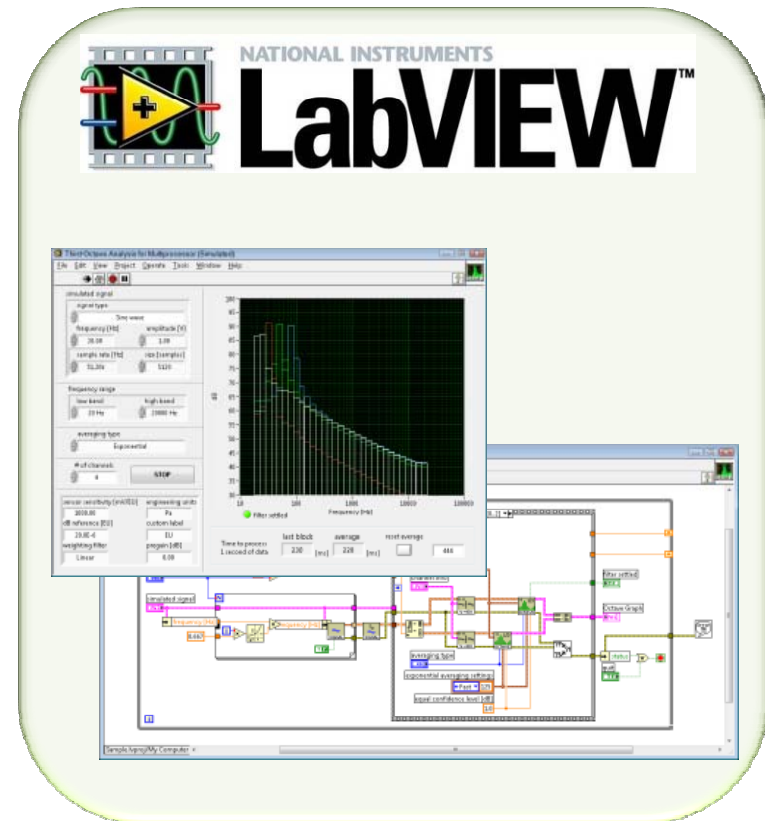
Benefícios de Negócios

- Multicore está possibilitando novas aplicações
- Aplicações embarcadas de tempo real e de alto desempenho podem ser desenvolvidas com multicore
- Para aplicações críticas, múltiplos núcleos podem significar um potencial aumento na produção

Introdução ao LabVIEW e ao Fluxo de Dados

Tópicos da Seção: LabVIEW e Fluxo de Dados

1. Introdução ao ambiente LabVIEW
2. Funções e Hierarquia
3. Fluxo de Dados e Paralelismo



O que é o LabVIEW?

- Linguagem de programação: programação gráfica baseada em fluxo de dados estruturado
- Ambiente de desenvolvimento completo: janela de gerenciamento de projeto, ferramentas de depuração, controle de código fonte e mais

Partes de uma Aplicação em LabVIEW

- O Diagrama de Blocos mostra graficamente o código fonte
- O Painel Frontal contém os itens da interface gráfica de usuário (GUI) (conectados ao código fonte)

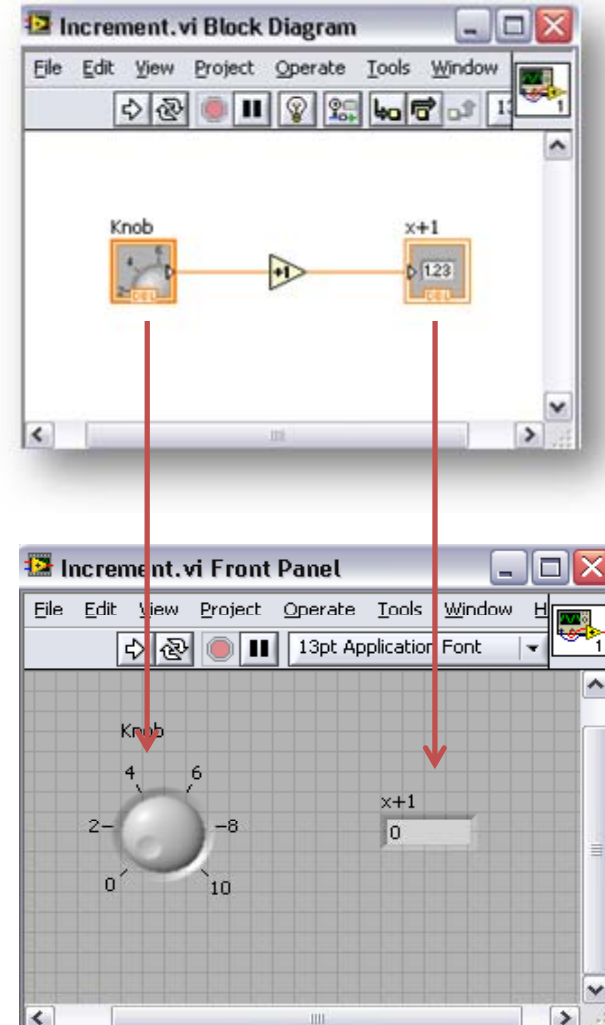
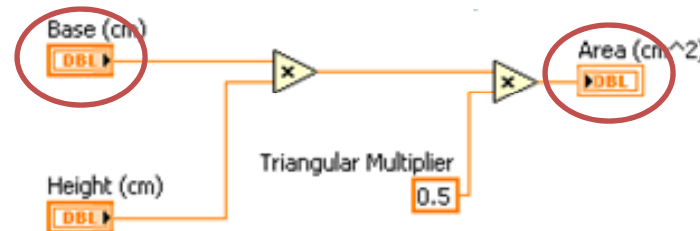
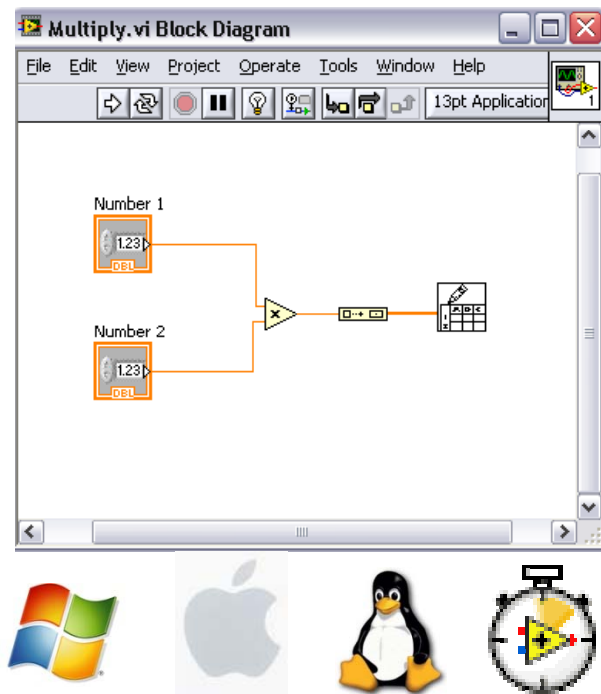


Diagrama de Blocos: Código Fonte

- O diagrama de blocos pode conter funções, estruturas e constantes
- Análogo a funções, subrotinas, elementos de controle de fluxo e constantes em linguagens de texto
- Os terminais representam entradas ou saídas

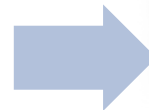


Programação Gráfica MultiPlataformas



0000001010101000000000000010100000101000000100100010
01010100100100110
00000000000000000000110100000000000000000000000000000000
0000000000000000000001100001110001100001100010000000000
10000110101111101111101111101111100000000000000000000000
00111111
0000000000000000000001100001100001110001100010000000100000
01101011111011111011111011111000000000000000000000000000
0000000110000000000000001000001100000000001111110000
0011000000
0000000011000100001100000000000000000000011001100000000000
000110000001000000001000000100000000100000100000000110
0000100010000000001000000010000010000000100000001000
00110000000011000000000100011101011000000000000000000000
110000000000000000000101110100101101100000010011100100
1110000000001010000011101100100000010100000111111001
0011011001110000010
0101010100111000000000101010100000000000000000000000101000

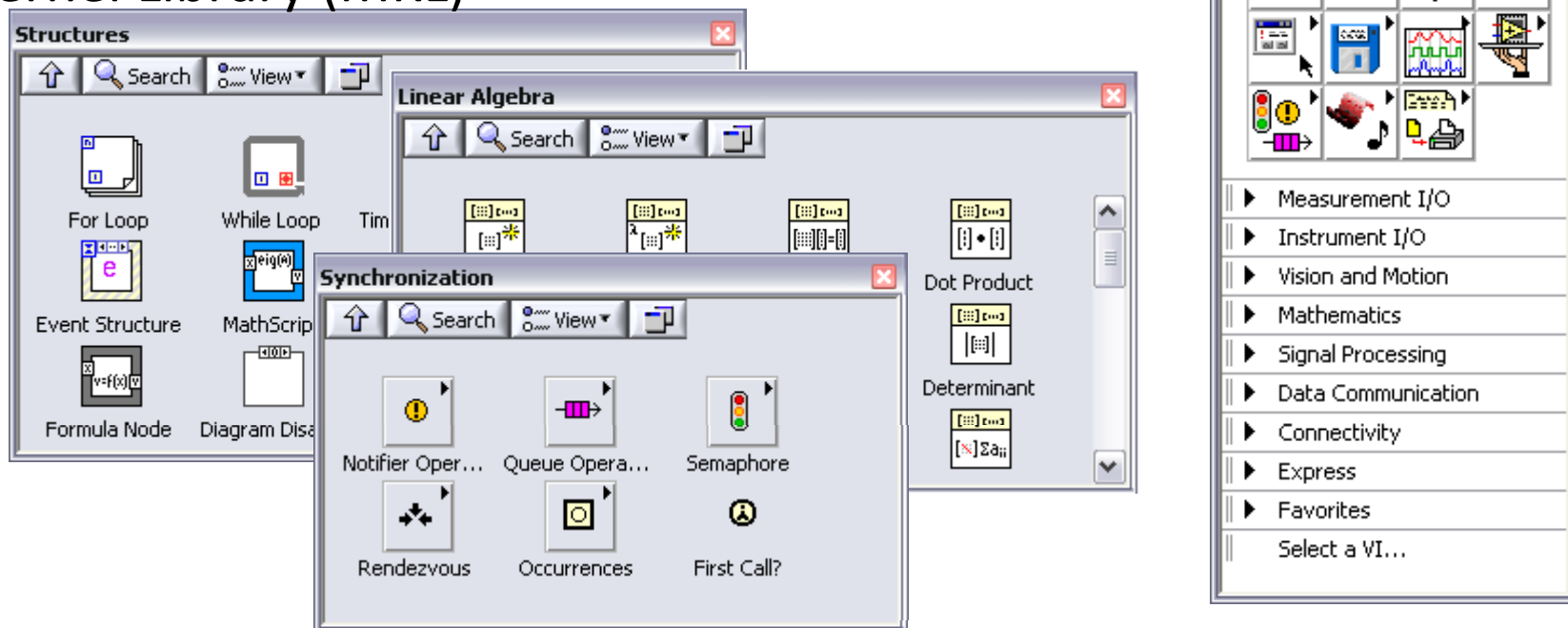
Código Fonte Gráfico



Código Compilado (binário)

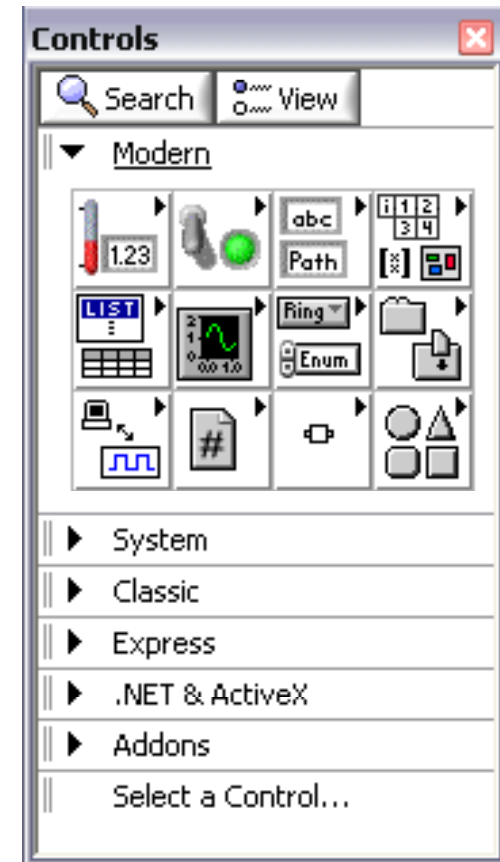
Diagrama de Blocos: Paleta de Funções

- Contém estruturas de programação e funções
- Contém IP especializado, incluindo bibliotecas matemáticas desenvolvidas em Intel Math Kernel Library (MKL)



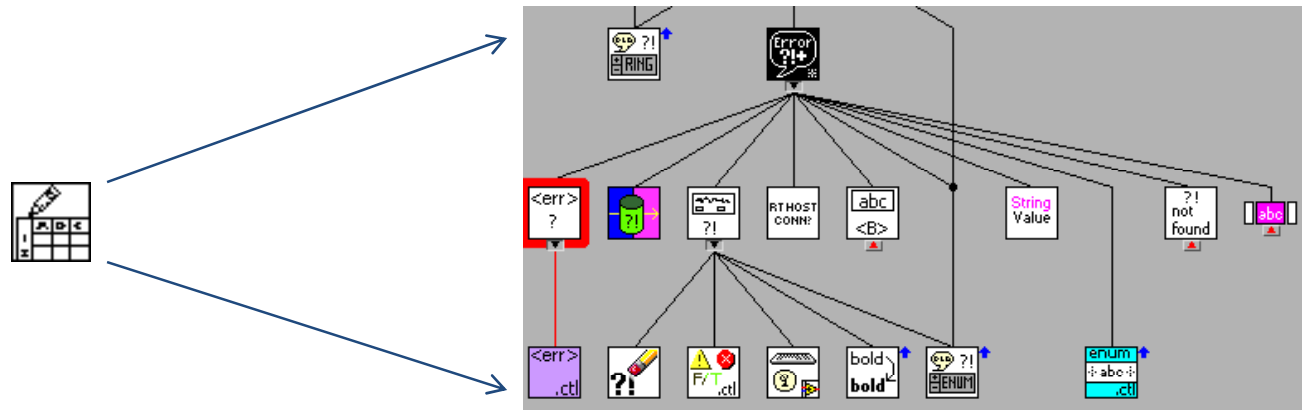
Painel Frontal: Paleta de Controles

- Controles (entradas): menus, caixas de texto, chaves, botões, etc.
- Indicadores (saídas): gráficos, LEDs, charts e outros mostradores



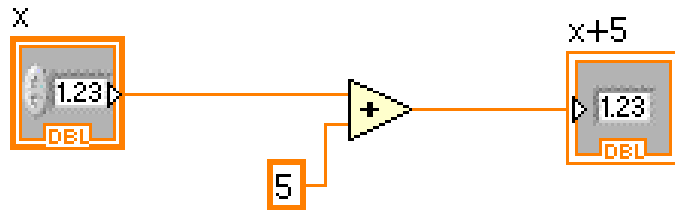
Hierarquia em LabVIEW

- Um subVI em LabVIEW é um VI que é chamado dentro de outro VI
- As entradas e saídas de um subVI (parâmetros) são expostos e acessados no diagrama de blocos
- Similar a funções e subrotinas em programação baseada em texto



O Paradigma do Fluxo de Dados

- A ordem de execução no LabVIEW é determinada pelo fluxo de dados
- Cada subVI ou função executa quando todas as entradas estão disponíveis, então produz os valores de saída
- Alto contraste com programação com fluxo sequencial



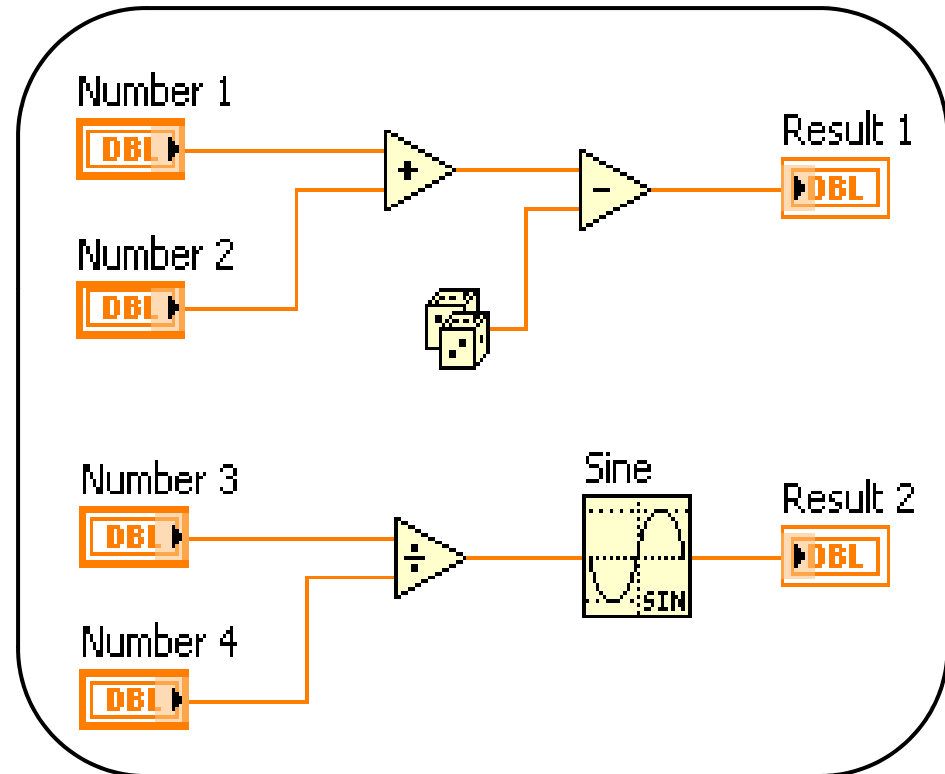
```
static int panelHandle;
static int plotColors[12] = {VAL_RED, VAL_GREEN, VAL_BLUE, VAL_CYAN,
VAL_MAGENTA, VAL_YELLOW, VAL_DK_RED, VAL_DK_BLUE, VAL_DK_GREEN,
VAL_DK_CYAN, VAL_DK_MAGENTA, VAL_DK_YELLOW};

int main(int argc, char *argv[])
{
    if( InitCVRTE(0,argv,0)==0 )
        return -1; /* out of memory */
    if( (panelHandle=LoadPanel(0,"Acq-IntClk-AnlgStart.uir",PANEL))<0 )
        return -1;
    SetCtrlAttribute(panelHandle,PANEL_DECORATION_BLUE,ATTR_FRAME_COLOR,VAL_BLUE);
    SetCtrlAttribute(panelHandle,PANEL_DECORATION_GREEN,ATTR_FRAME_COLOR,VAL_GREEN);
    SetCtrlAttribute(panelHandle,PANEL_DECORATION_YELLOW,ATTR_FRAME_COLOR,VAL_YELLOW);
    NIDAQmx_NewPhysChanAICtrl(panelHandle,PANEL_CHANNEL,1);
    DisplayPanel(panelHandle);
    RunUserInterface();
    DiscardPanel(panelHandle);
    return 0;
}
```

Desafio sobre Fluxo de Dados

Qual executa primeiro?

- a) Adição
- b) Subtração
- c) Número Aleatório
- d) Divisão
- e) Seno



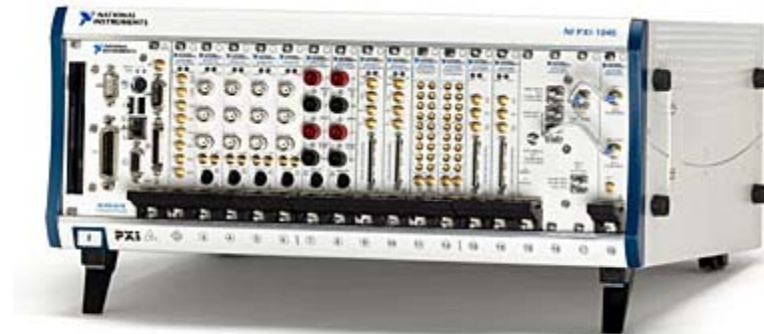
Fluxo de Dados e Execução Paralela

- Programação com fluxo de dados significa paralelismo intrínseco em sua aplicação
- Seções de código paralelo em um programa com fluxo de dados podem ser facilmente visualizadas

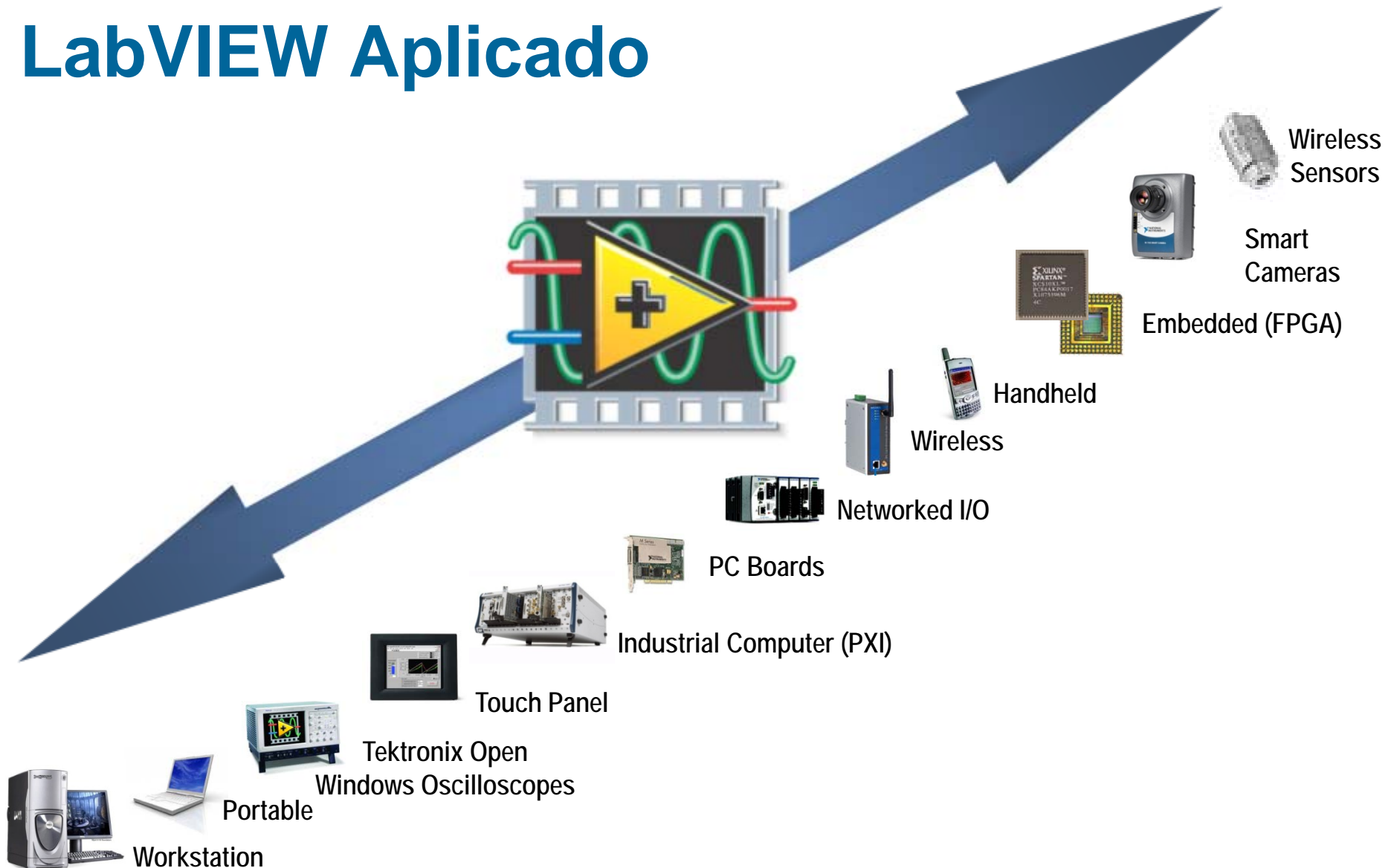
Ponto Chave: Fluxo de dados é uma tecnologia que torna possível a programação de sistemas com múltiplos processadores

Forte Integração com Hardware

- O LabVIEW pode ser programado para realizar aquisição de dados de E/S modulares, câmeras, sensores e outras fontes
- VIs Expressos rapidamente conduzem usuários através do processo de coletar e analisar dados

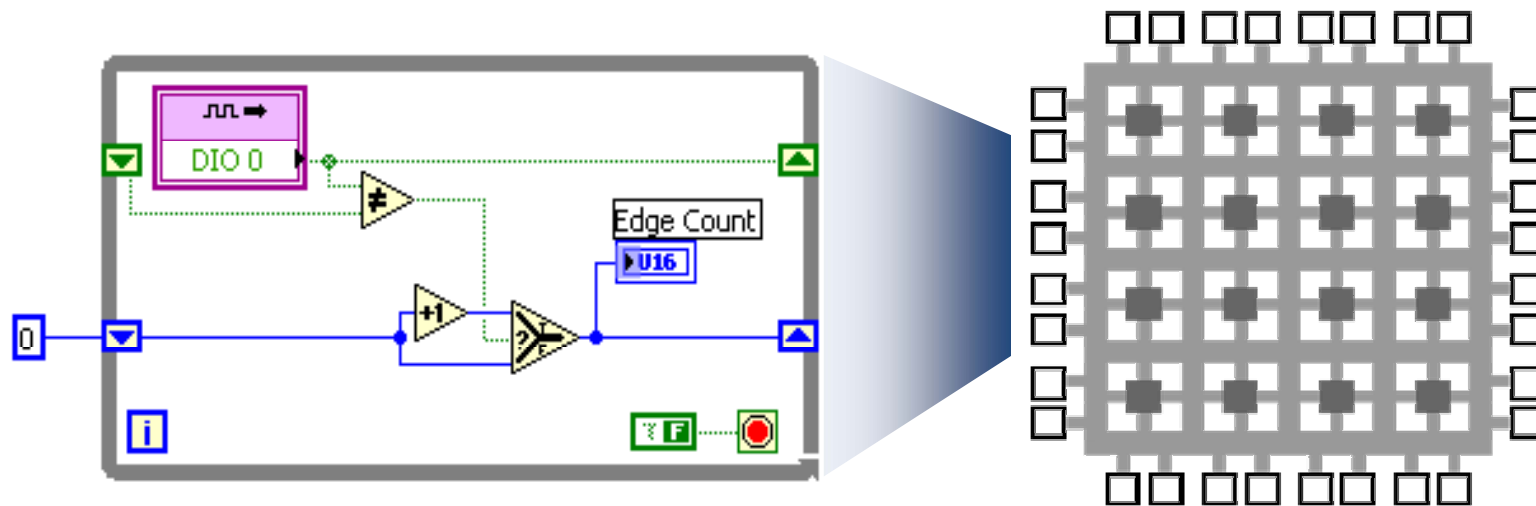


LabVIEW Aplicado



Extendendo o Paradigma de Fluxo de Dados

- Programação de hardware paralelo com o LabVIEW FPGA
- Mesma abordagem de se programar CPUs, sendo que o código executa diretamente no hardware



Multithreading

Tópicos da Sessão: Multithreading

- Introdução às threads
- Multithreading em LabVIEW
- Agendamento do Sistema Operacional (OS)
- Afinidade do processador



O Que São Processos e Threads?

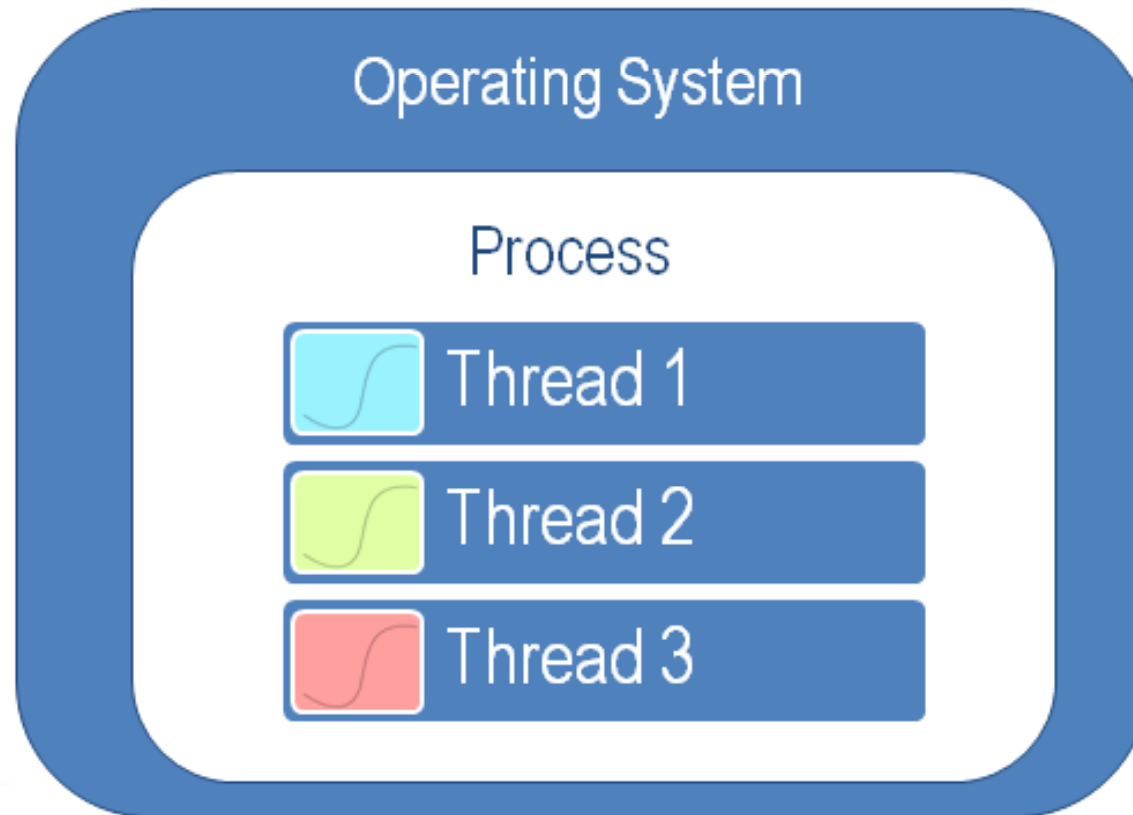
- Processos

- Todo program executa em um processo
- Processos disponibilizam recursos necessários para executar (memória, manipulação de arquivos, sockets, janelas)

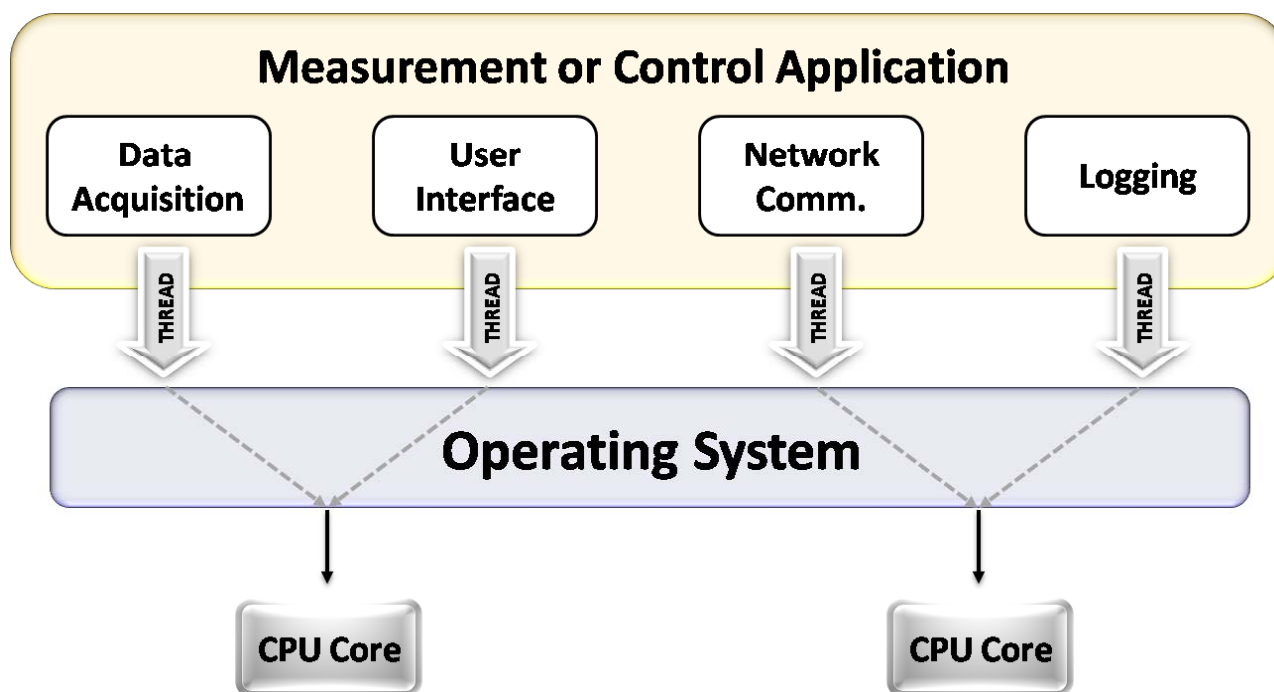
- Threads

- Threads são entidades dentro de um processo que podem ser executadas (frequentemente em paralelo)
- Todas as threads compartilham recursos do processo

O Que São Processos e Threads?



Criando Aplicações Multithreaded



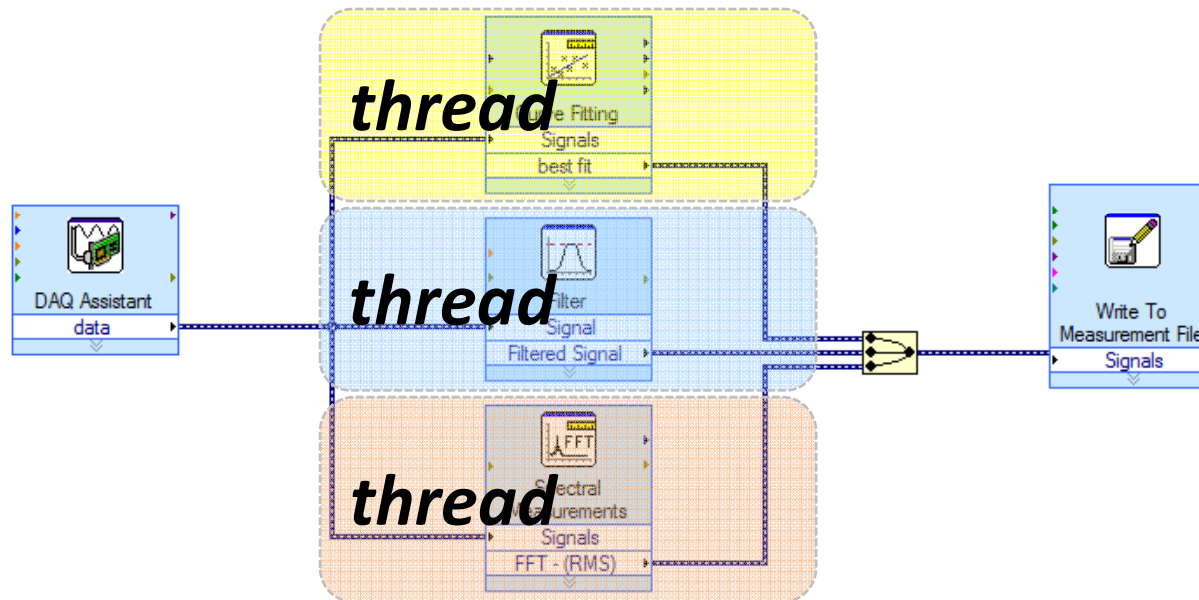
As Aplicações **devem** fazer uso de threads para se beneficiar dos processadores multicore.

Dividindo Programas em Threads no LabVIEW

- Threading automático
 - O algoritmo de clumping do LabVIEW designa automaticamente código para as threads baseado no paralelismo
 - Muitos dos programas existentes irão rodar mais rápido em um sistema Multicore **sem qualquer alteração**
- Threading manual
 - Force seções de código a executarem em uma thread

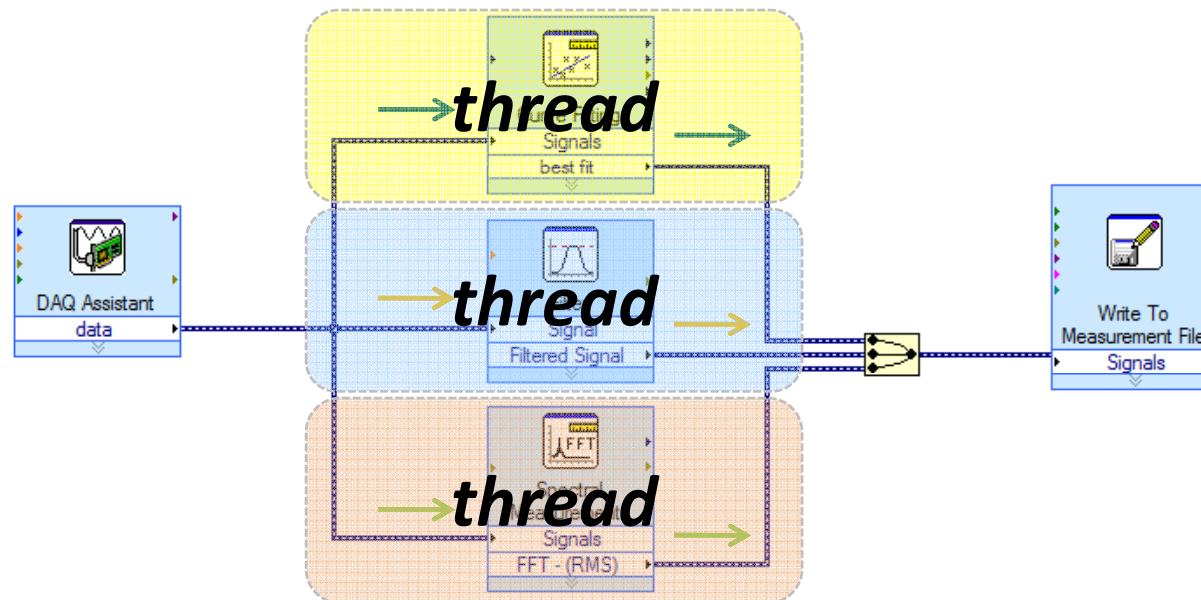
Multithreading Automático

- LabVIEW divide um programa em múltiplos threads (originalmente introduzido em 1998 com LabVIEW 5.0)
- Simplificação mostrada abaixo; caminho com código paralelo executa em threads separados para rodar em hardware paralelo



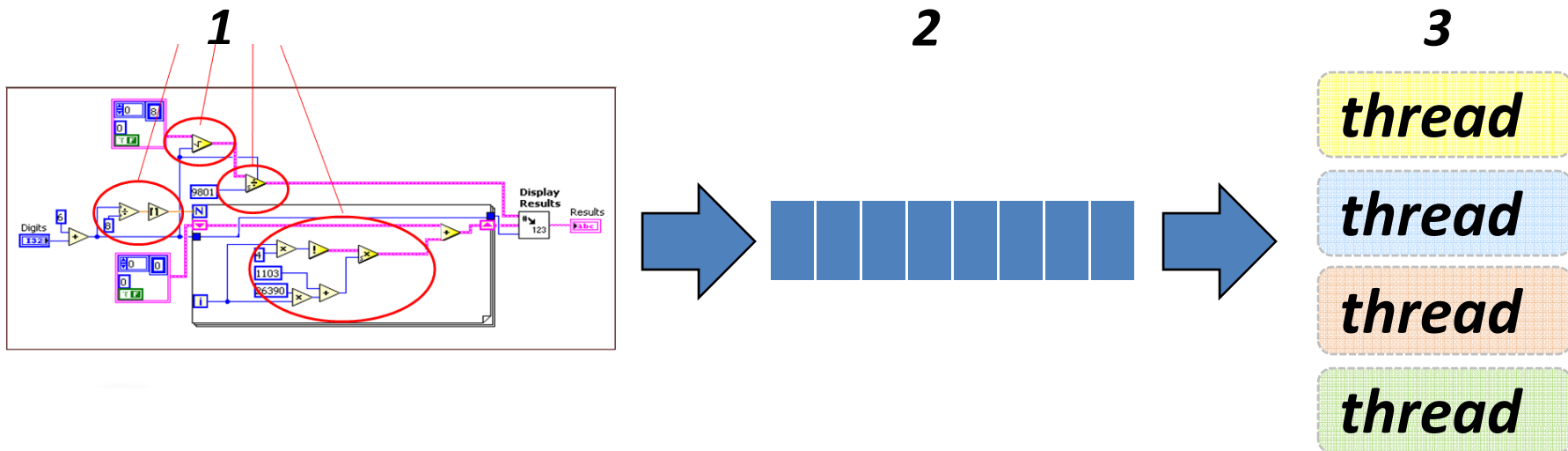
Multithread Automático em LabVIEW

- LabVIEW automaticamente divide cada aplicação em múltiplas threads de execução



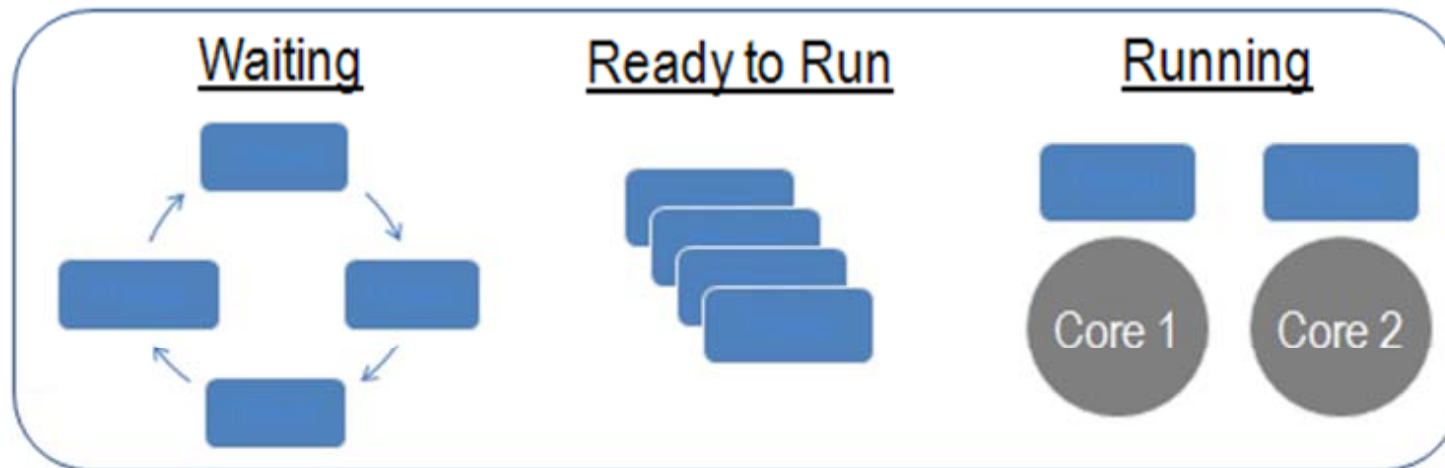
Multithreading em LabVIEW – Em Detalhes

1. Instruções de código sequencial e dependências são agrupadas
2. Informações sobre quais partes de código podem rodar juntas são armazenadas em uma fila de execução
3. Se houver paralelismo suficiente no diagrama de blocos, ele irá executar simultaneamente em todos os threads do sistema



Como o OS Agenda e Roda Threads

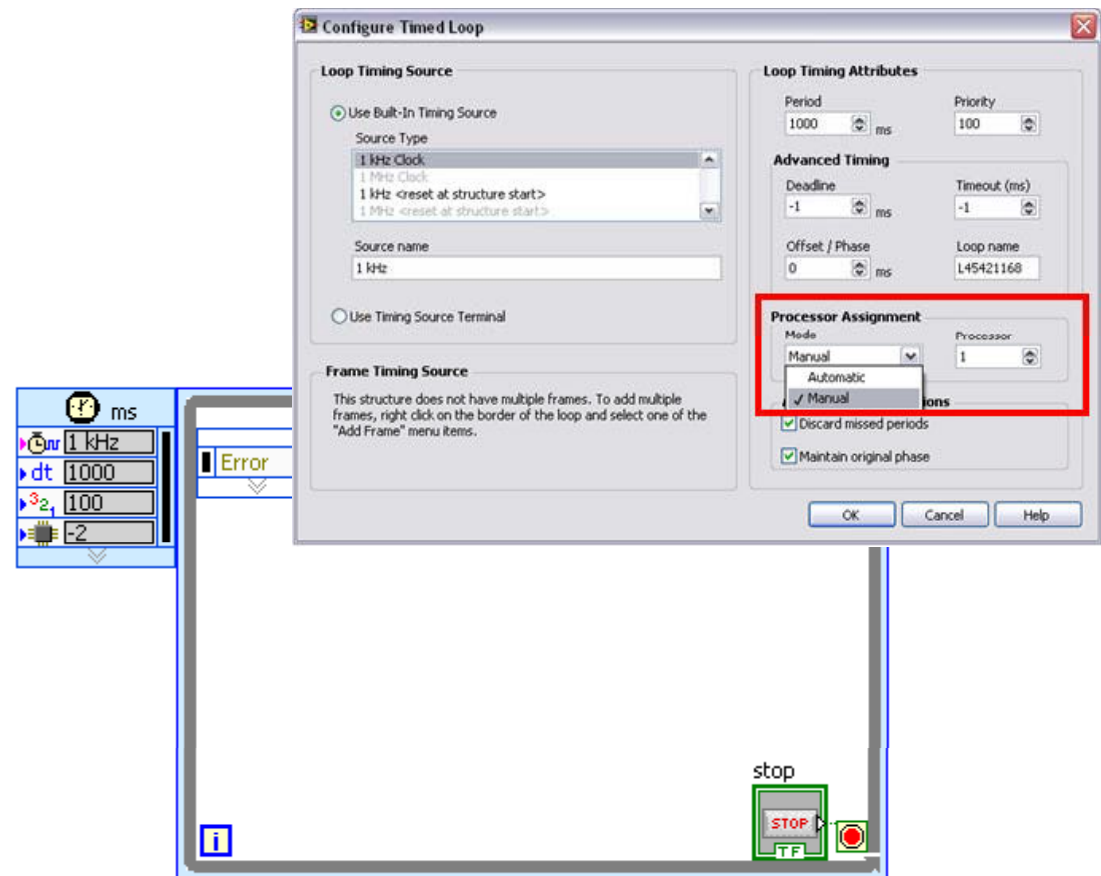
- Monitoramento das threads em um processo e seus estados
- Balanceamento de diferentes threads em diferentes núcleos de processamento quando possível



Exemplo de alguns estados de threads

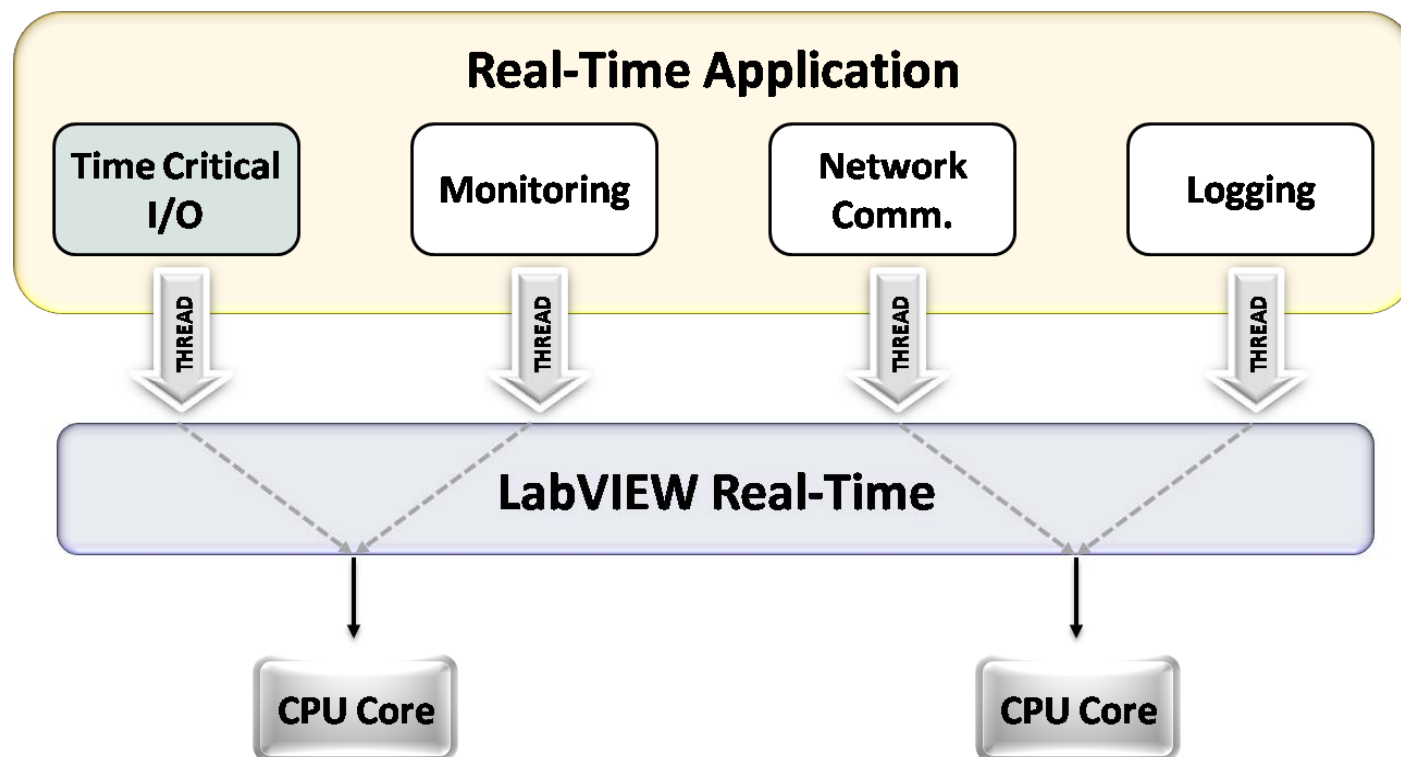
Threading Manual com Timed Loops

- Código dentro de Timed Loops executa em um único thread
- Threads podem ser classificados com uma prioridade relativa
- Configurar afinidade do processador



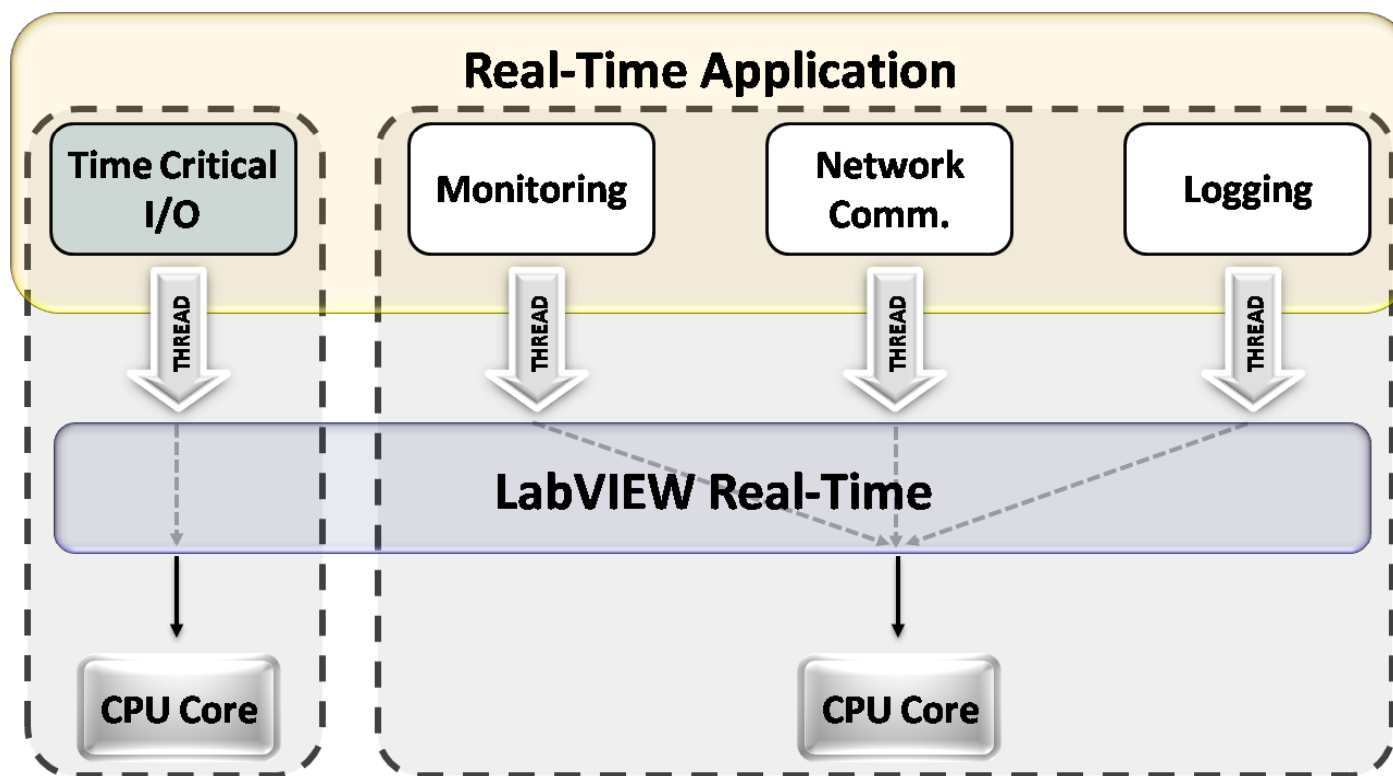
Sistemas Determinísticos de Tempo Real

LabVIEW 8.5 adicionou Symmetric Multiprocessing (SMP) para sistemas de tempo real.



Controle da Execução da CPU para Tempo Real

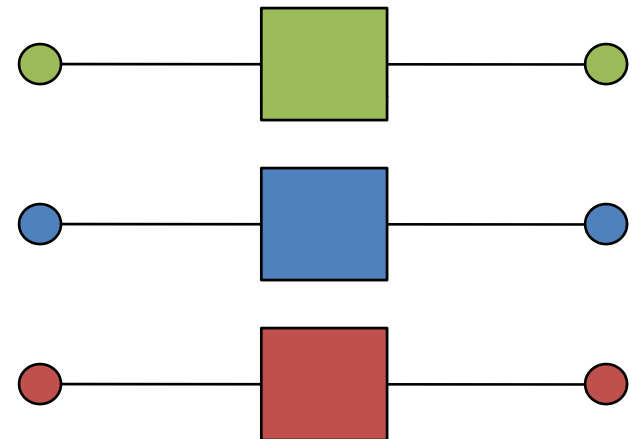
Em LabVIEW 8.5, os usuários podem designar código para um núcleo do processador específico usando o Timed Loop



Técnicas de Programação Paralela

Tópicos da Seção: Técnicas de Paralelismo

- Paralelismo de tarefas e multithreading automático
- Paralelismo de dados
- Pipelining e balanceamento de estágio
- Lei de Amdahl

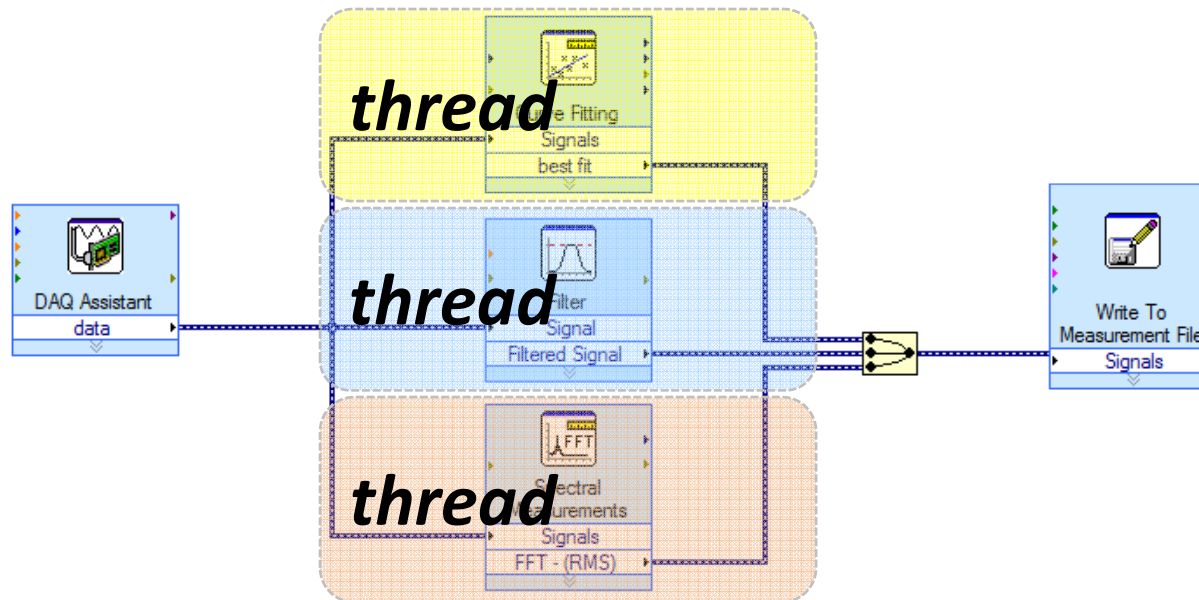


Paralelismo de Tarefas

- 1) Procurar tarefas que podem rodar em paralelo
- 2) Planejar o código para refletir esse paralelismo
 - Eliminar dependência de dados
 - LabVIEW automaticamente identifica código paralelo que pode ser dividido em múltiplos threads!

Exemplo: Paralelismo de Tarefas

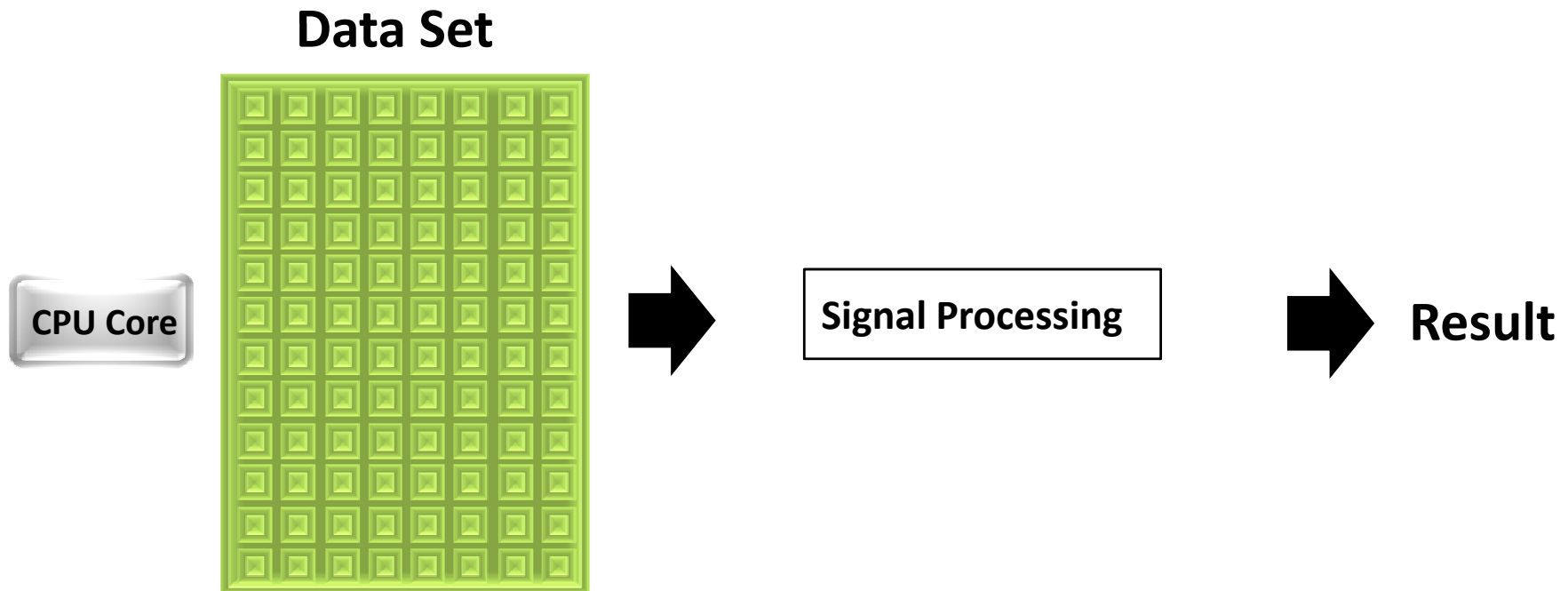
Operações Paralelas



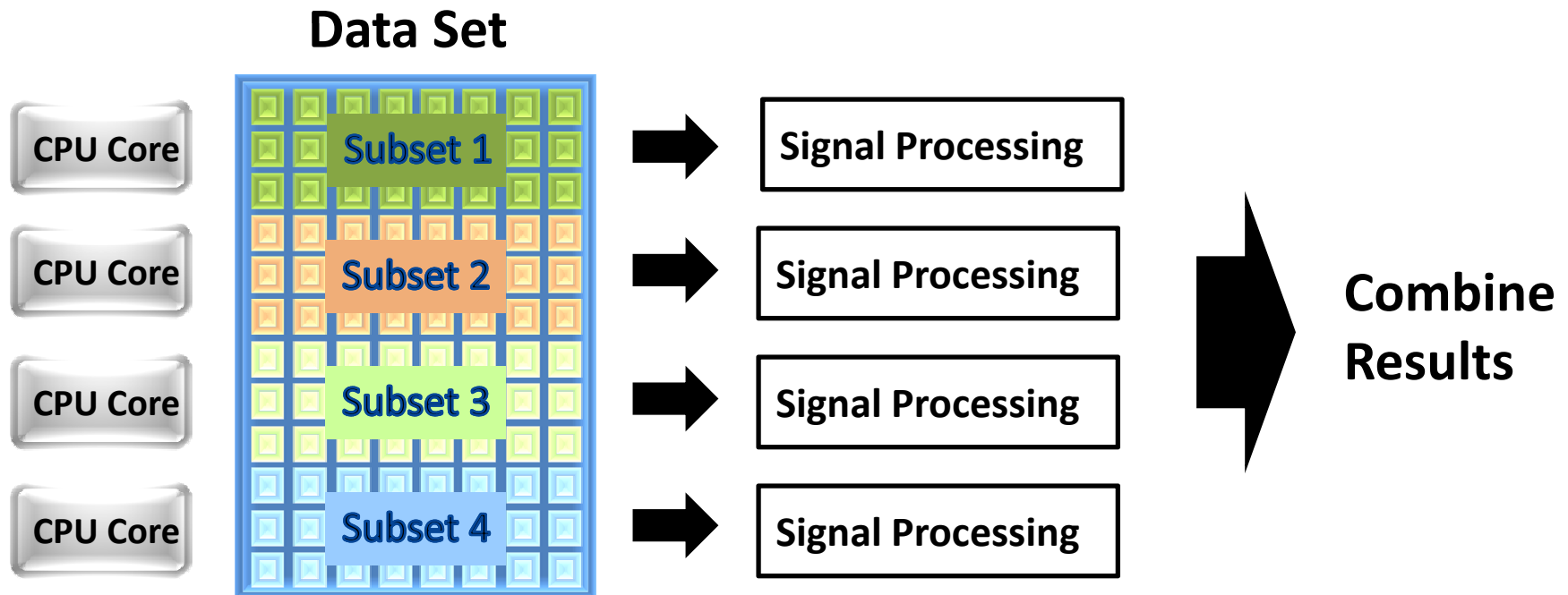
Paralelismo de Dados

- 1) Procurar por um grande conjunto de dados que pode ser processado em dois ou mais “pacotes” independentemente
- 2) Planejar o código:
 - Dividir os dados
 - Processar os dados em paralelo
 - Combinar os resultados individuais em um resultado geral

Exemplo: Paralelismo de Dados

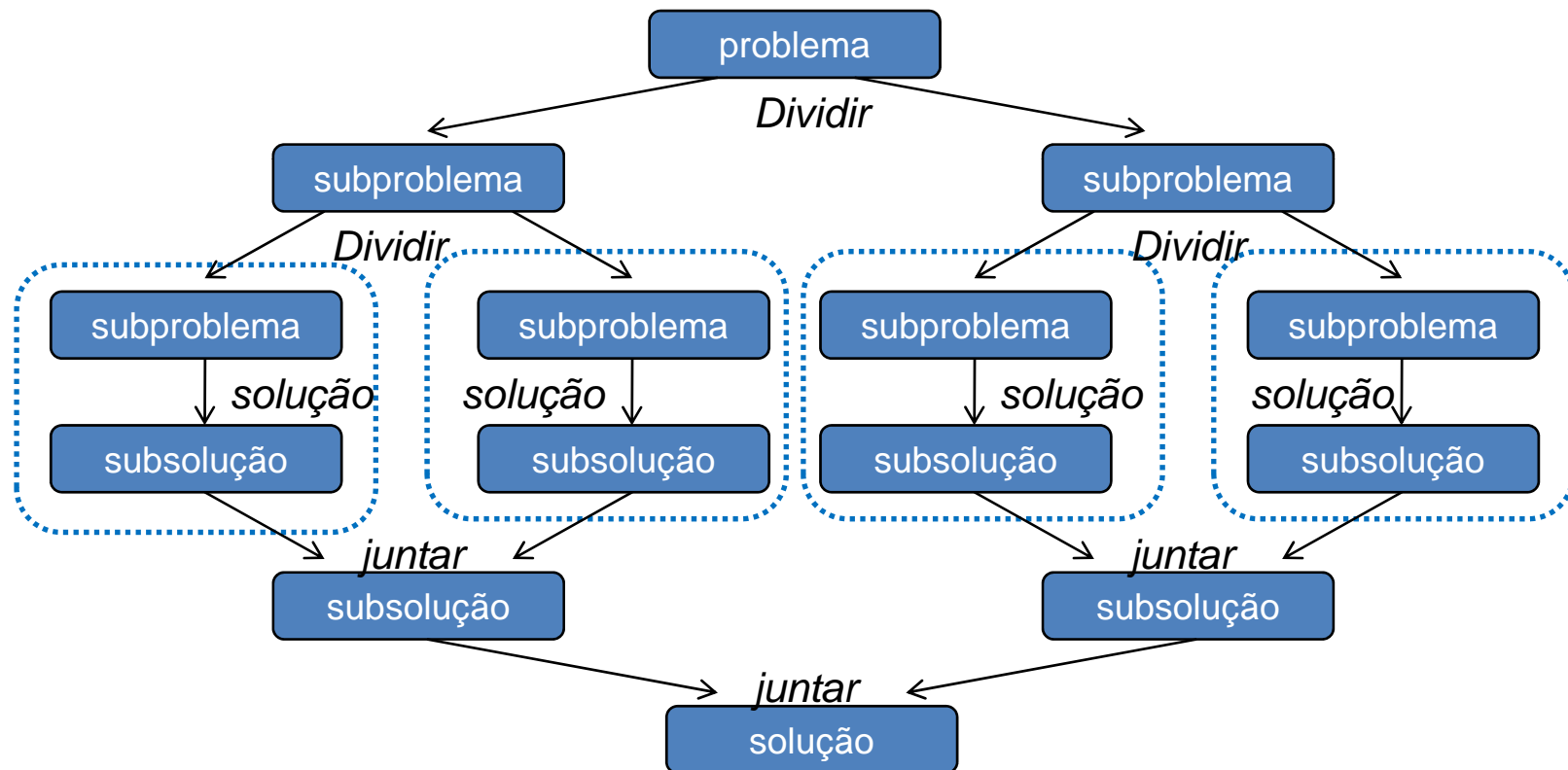


Exemplo: Paralelismo de Dados



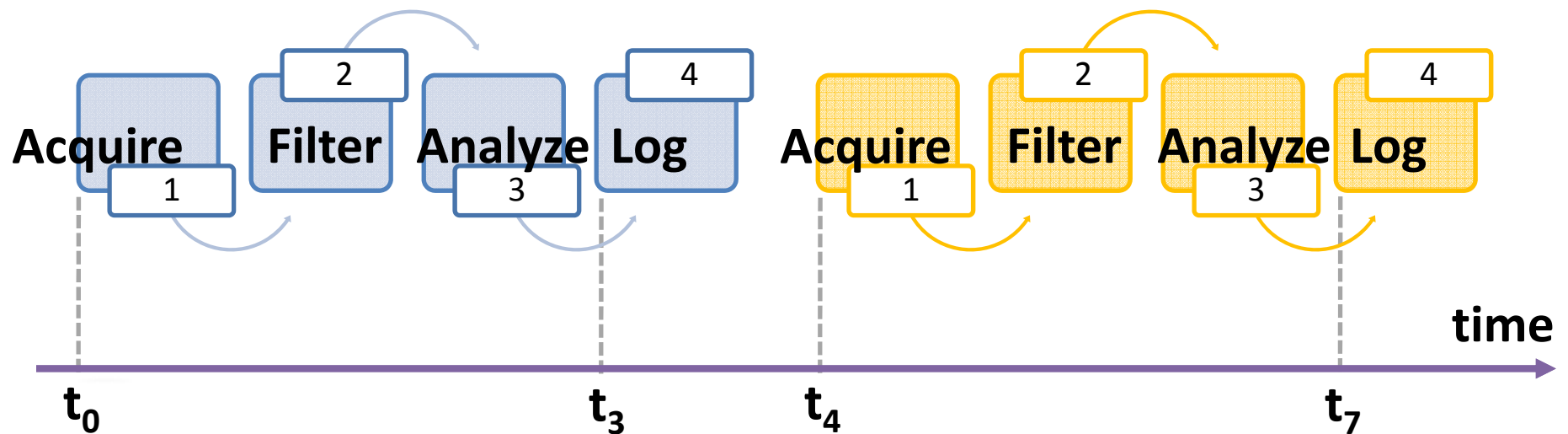
Paralelismo de Dados: Dividir & Conquistar

- Útil para algoritmos recursivos, processamento de imagens, etc

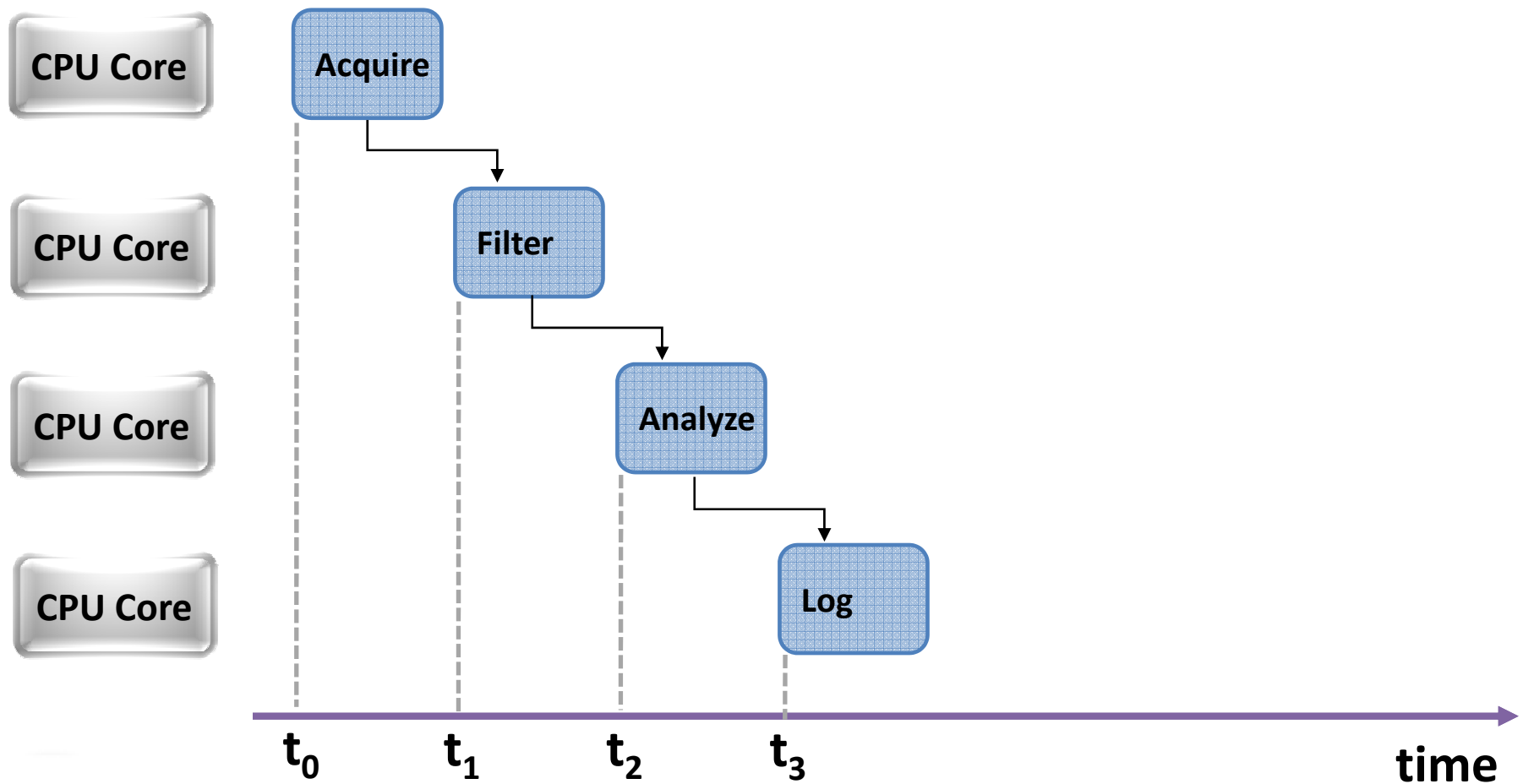


Pipelining

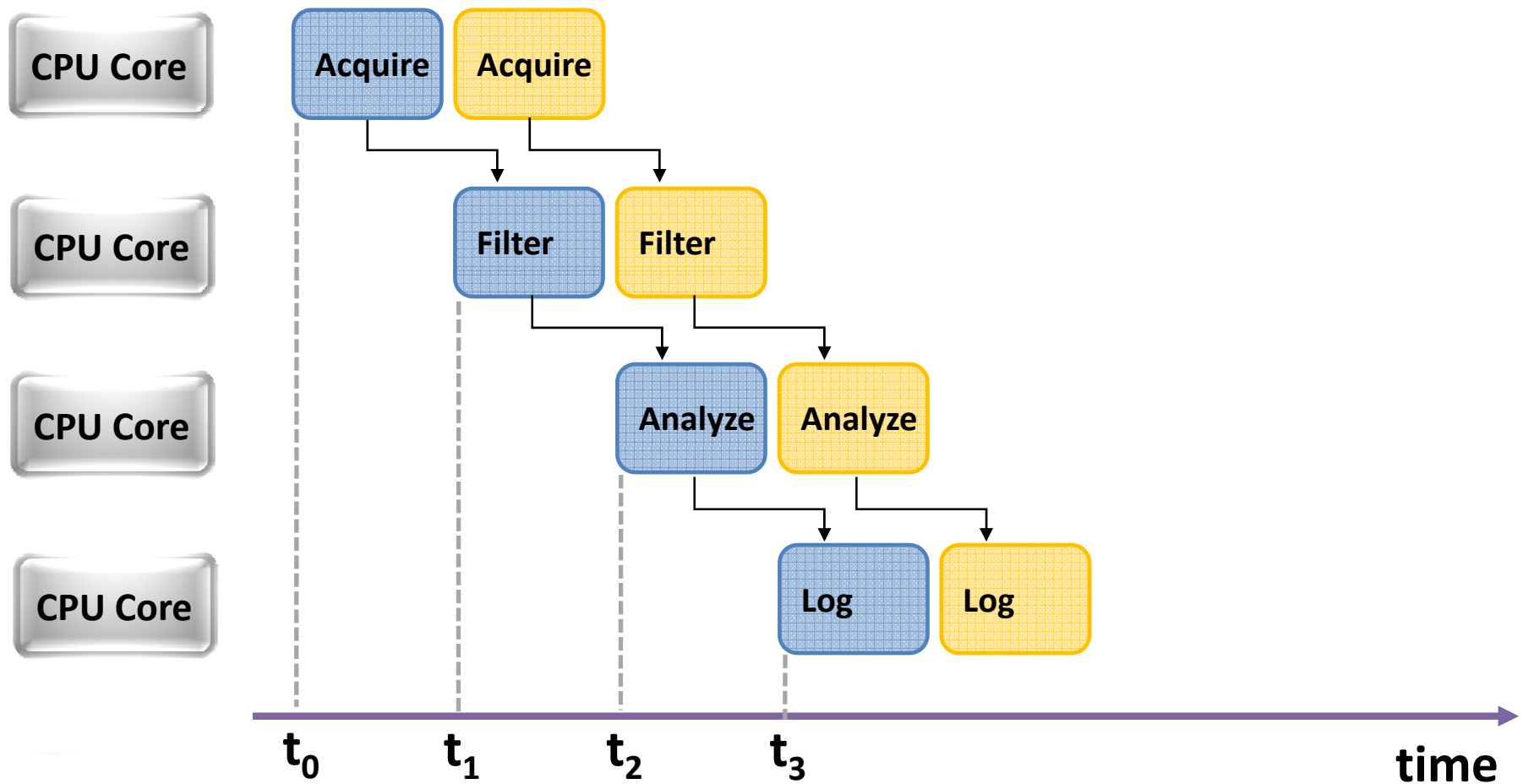
- Motivação: muitos programas contêm algoritmos com múltiplos passos sequenciais
- Aplicar pipelining pode aumentar a produtividade



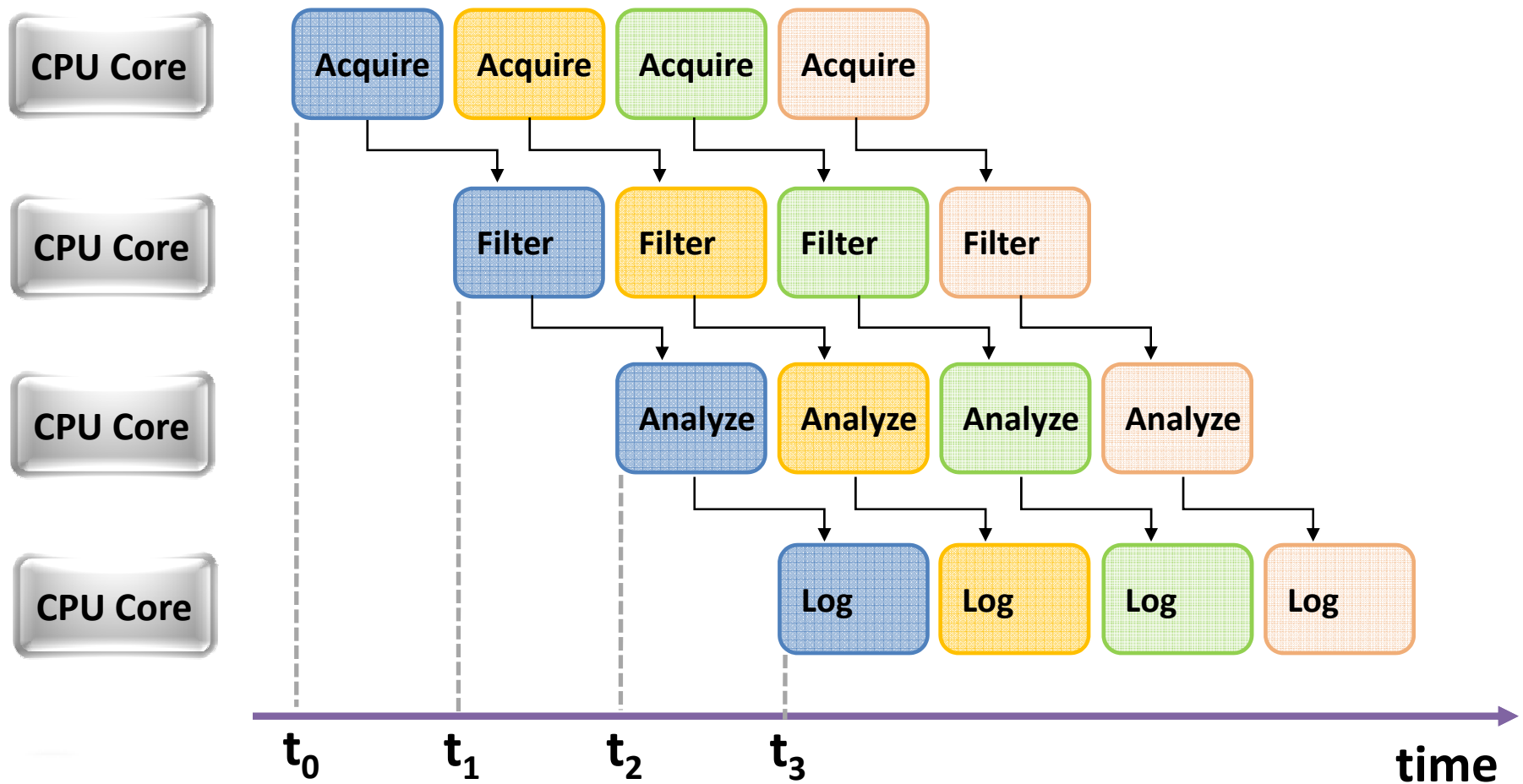
Estratégia Pipelining



Estratégia Pipelining

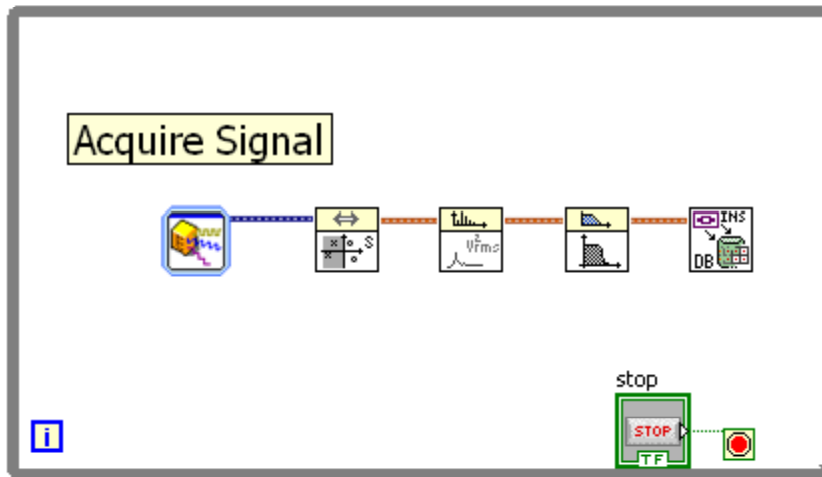


Estratégia Pipelining

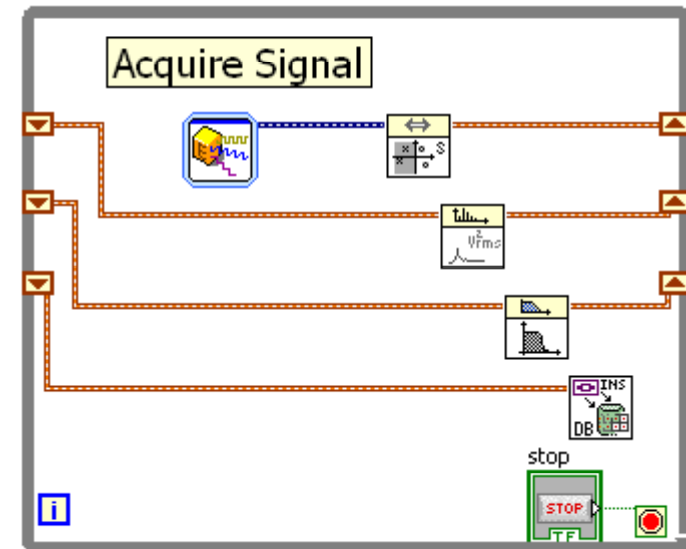


Pipelining em LabVIEW

Sequential



Pipelined

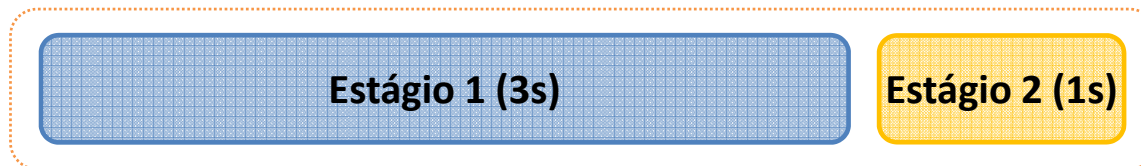


Nota: existem outras técnicas para pipelining tal como usar múltiplos loops com filas e usar feedback nodes

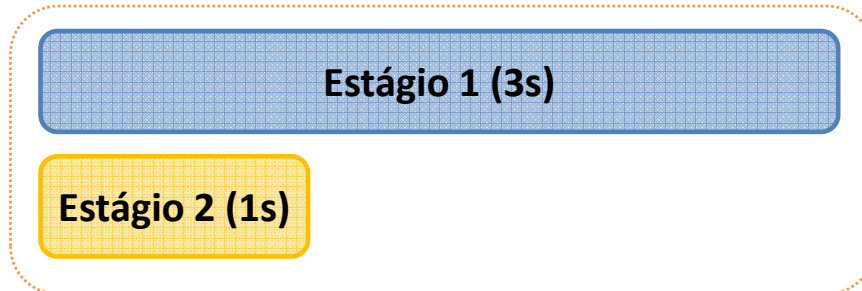
Balanceando Estágios Pipeline

- Caminho crítico é o maior estágio pipeline
- Pipelining com estágios desbalanceados podem não garantir um aumento significativo de desempenho

Sem Pipeline (tempo total = 4s)



Com Pipeline (tempo total = 3s): Melhoria = 1.33X (não ideal para pipelining)



Previendo o Aumento de Velocidade – Lei de Amdahl

- Pode ser usada para calcular o aumento de velocidade esperado de uma aplicação após paralelizar uma seção

$$S_{total} = \frac{1}{\sum_{k=0}^n \left(\frac{P_k}{S_k} \right)}$$

Ponto Chave: invista tempo na aplicação do paralelismo e na otimização de sessões de código que demoram mais para executar

Sumário: Estratégias de Paralelismo

- Paralelismo de Tarefas
 - Melhor para programas com funcionalidades independentes que possuem pouca dependência ou compartilhamento de dados
- Paralelismo de Dados
 - Melhor para aplicações que lidam com um conjunto grande de dados que pode ser dividido e manipulado independentemente
- Pipelining
 - Melhor para código que repete vários estágios sequenciais com dependência de dados entre eles

Desafios Multicore e Depuração

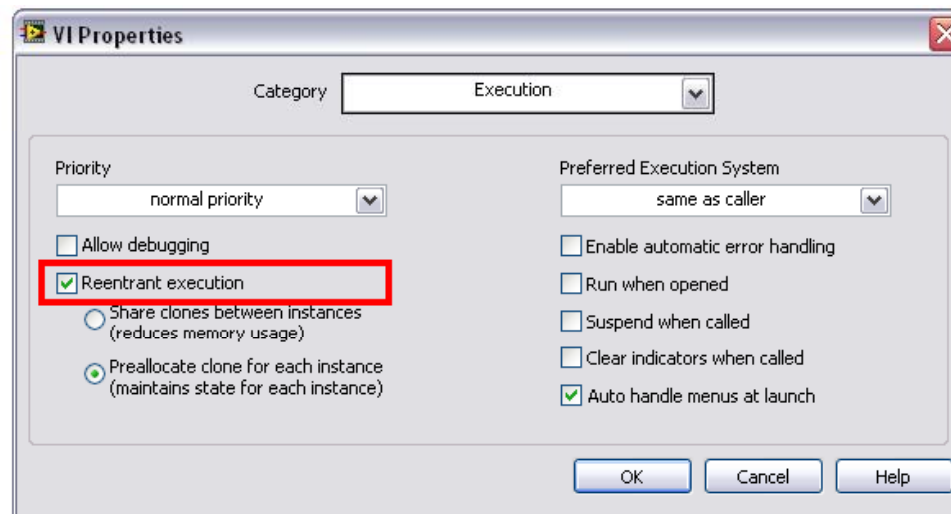
Tópicos da Sessão:

Desafios Multicore e Depuração

- Reentrância e execução destacada
- Sincronização de threads
- Rastreamento da execução
- Analisadores de desempenho
- Considerações sobre transferência de dados

Reentrância

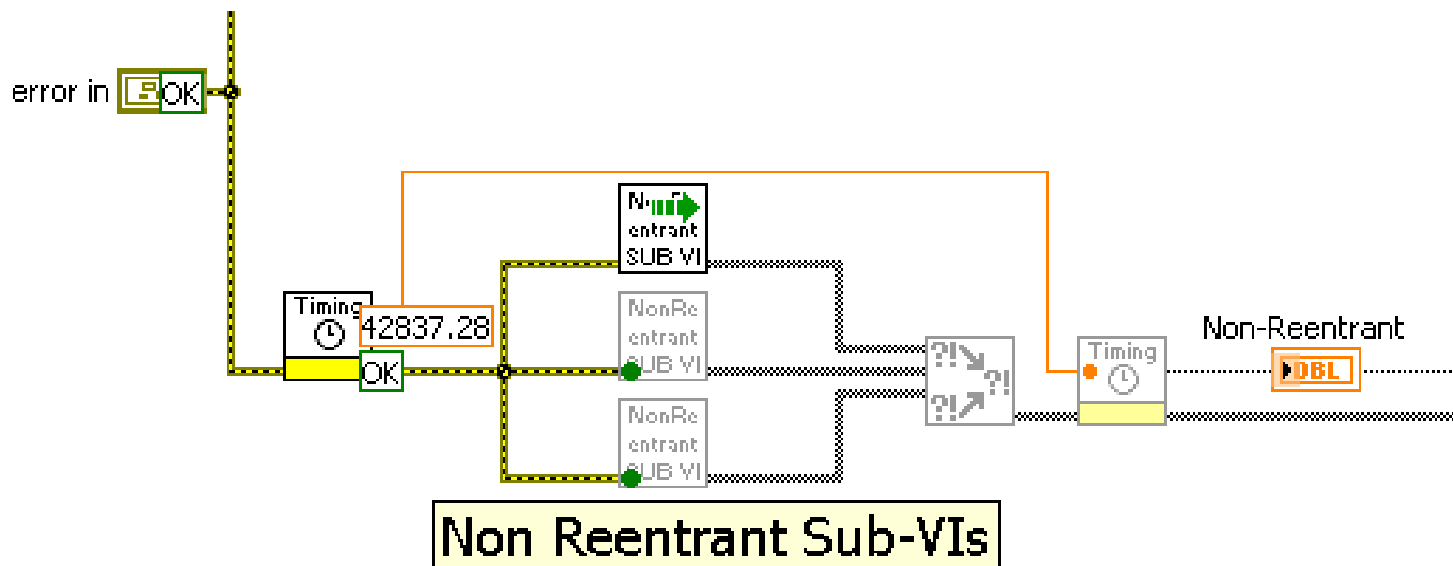
- Todos os VIs em LabVIEW podem ser configurados para serem reentrantes
- Permite que cada subVI use espaços de memória separados



Todos os subVIs que serão executados múltiplas vezes em paralelo precisam ser configurados como reentrantes

Execução Destacada

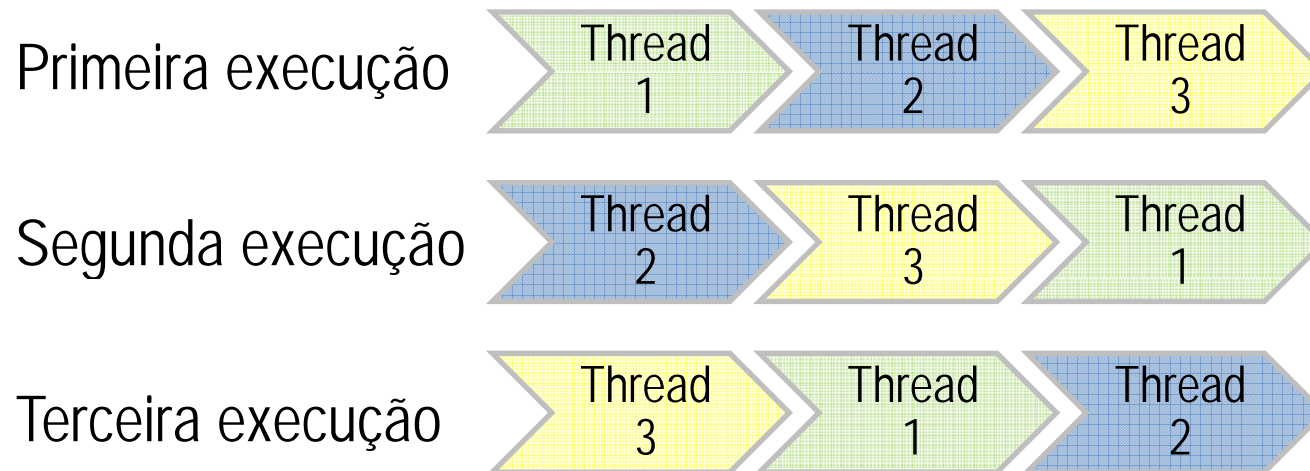
- Visualize sessões paralelas do código durante a execução
- Nota: essa ferramenta torna a execução lenta, então lembre-se de desligá-la quando terminar a depuração



Demonstração: Execução Reentrante

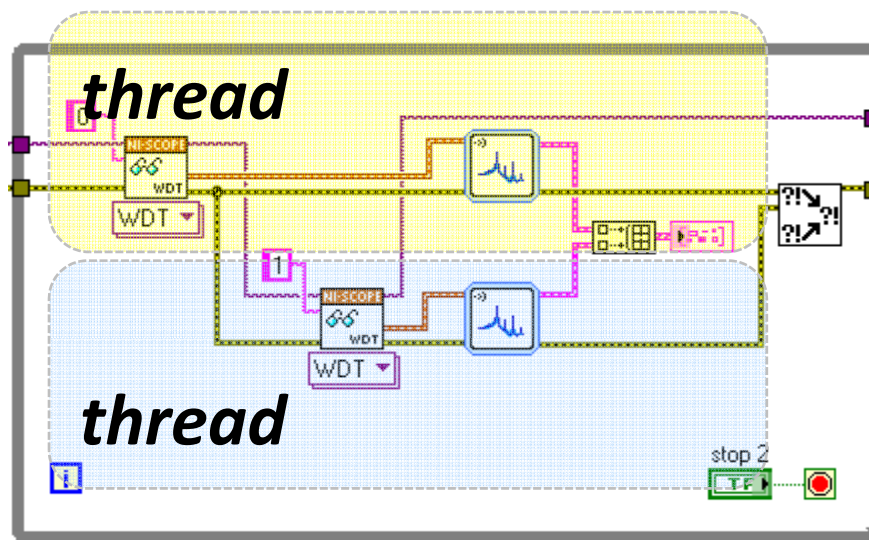
Sincronização de Threads

- Nenhuma garantia que o OS irá agendar threads na sequencia correta sem métodos de sincronização
- A ordem de eventos pode mudar a cada execução devido ao agendamento das threads



Sincronização de Código com Fluxo de Dados

- Paradigma do fluxo de dados garante sincronização

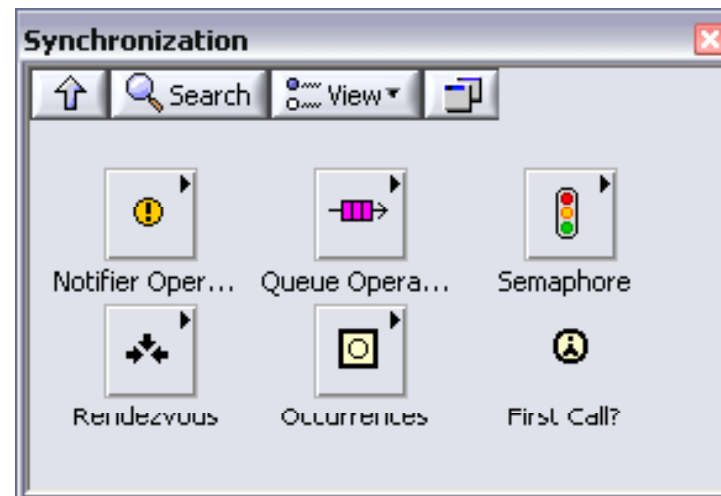


Caminhos paralelos do código são sincronizados e a ordem de execução é determinada pelos fios do LabVIEW

Fluxo de dados é o ponto chave que possibilita programação multicore

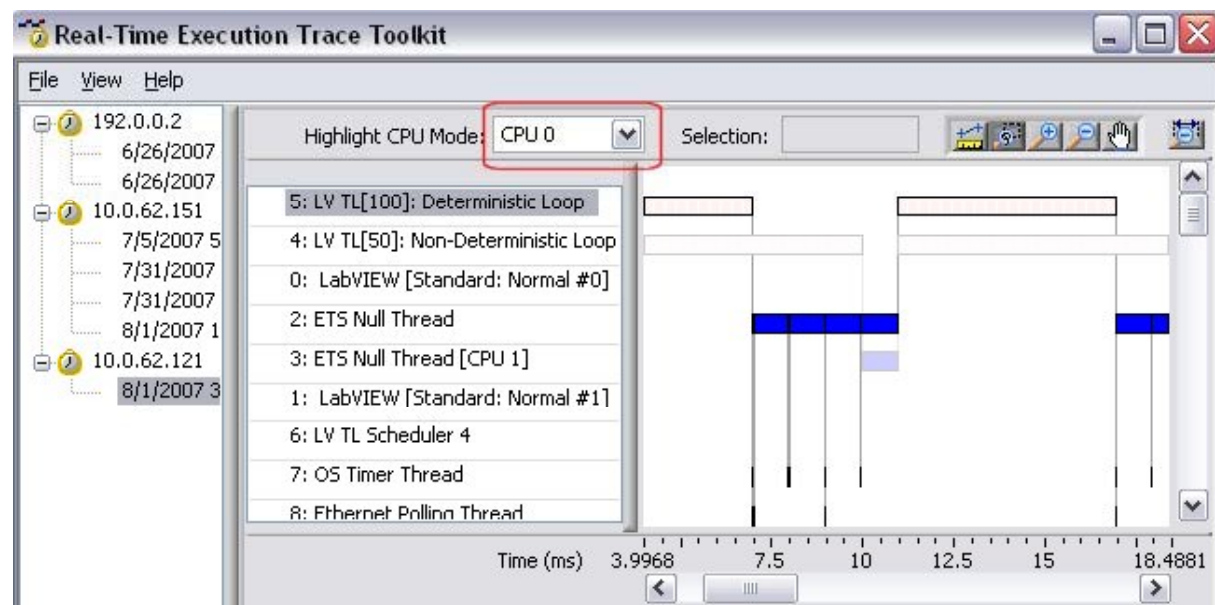
Sincronização em LabVIEW

- Quando sincronização de baixo nível é necessária, use mecanismos de sincronização que incluem:
 - Queues
 - Notifiers
 - Semaphores
 - Rendezvous
 - Occurrences



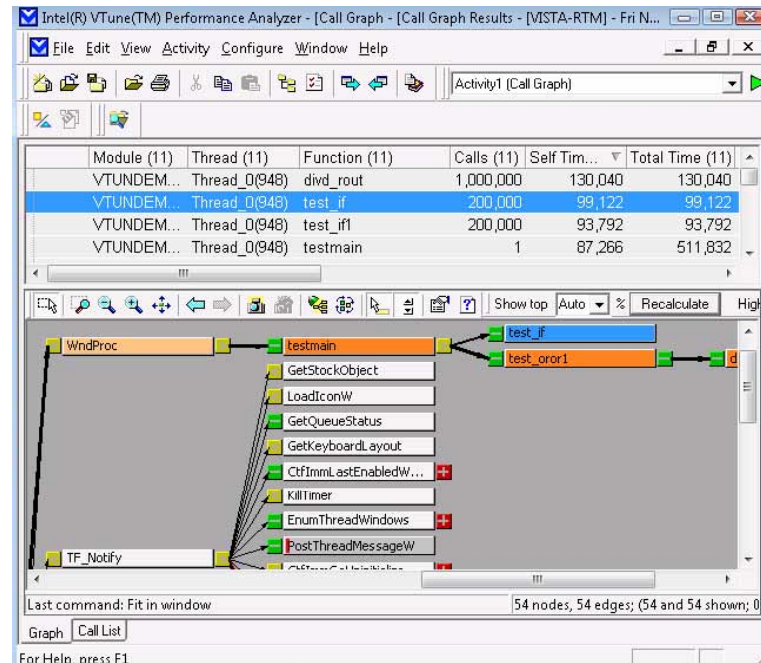
Rastreamento da Execução

- Em sistemas de tempo real, depuração rastreada pode mostrar a atividade das threads no SO
- Atividade das threads em cada núcleo é mostrada selecionando uma CPU particular



Analísadores de Desempenho

- Analísadores de desempenho fornecem informação detalhada do sistema (uso CPU, uso de memória e sucessos/perdas cache)
- LabVIEW pode chamar em Windows analisadores de desempenho
- Exemplos:
 - Windows Perfmon
 - Intel's VTune

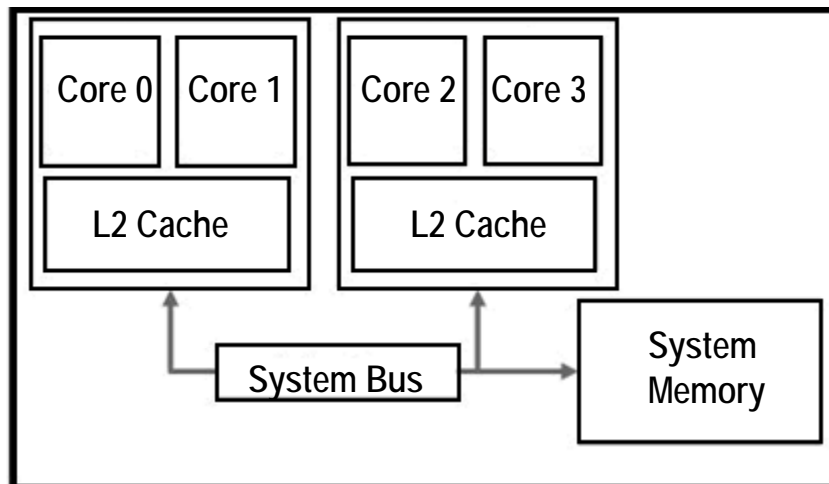


Considerações sobre Transferência de Dados

- Considere cuidadosamente a arquitetura de memória do sistema quando programando para Multicore
- Evite transferir grandes conjuntos de dados entre núcleos (relativo ao tamanho do cache)
- Transferências de memória muito pequenas podem causar repetidas atualizações na coerência do cache

Layout Físico do Processador

- Distância entre processadores e qualidade das conexões do processador podem ter um grande efeito
- Transferir dados no barramento do sistema é muito mais lento que acessar uma cache compartilhada



*Exemplo Cache
Compartilhada:
Dual-Core, Dual
Processor System*

Fonte: Tian and Shih, Software Techniques for Shared-Cache Multi-Core Systems, Intel Software Network

Projetando sua Aplicação Multicore

Tópicos da Sessão:

Projetando sua Aplicação Multicore

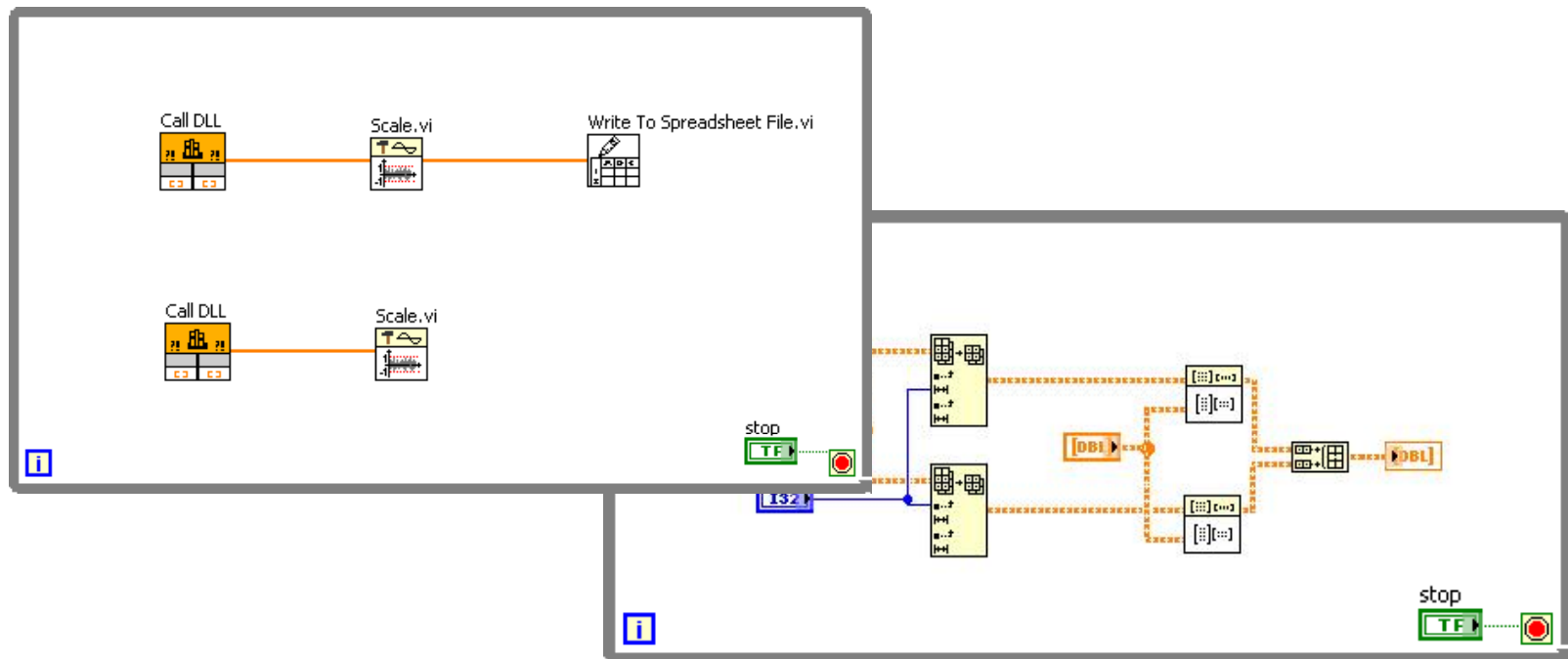
- Padrões Comuns de Projeto:
 - Loop simples com técnicas de programação paralela
 - Múltiplos loops independentes
 - Produtor/consumidor (transferir dados com filas)

Padrão de Projeto

Técnicas de Paralelismo

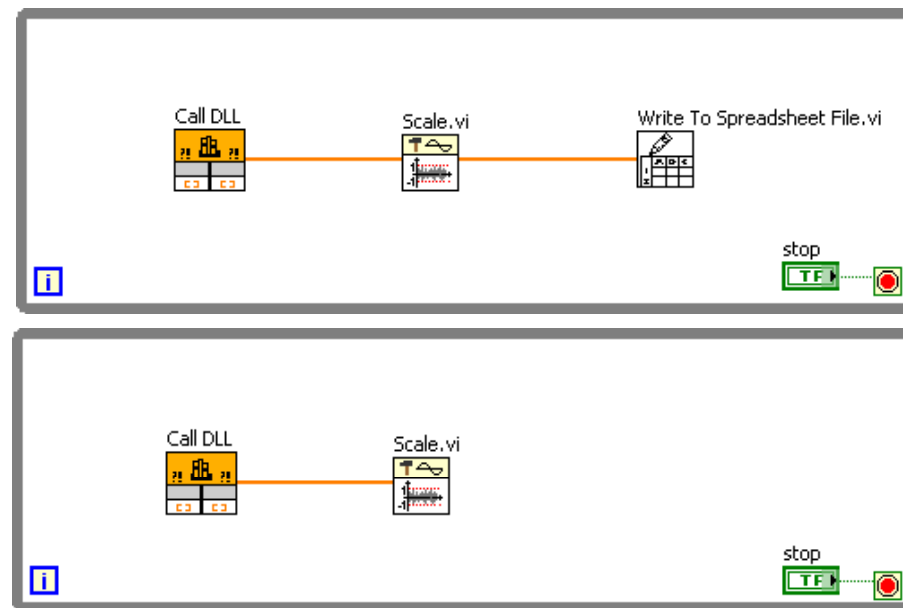
Padrões de Loop Simples

- Use técnicas de paralelismo (paralelismo de tarefas, paralelismo de dados, pipelining) dentro de um loop
- Velocidade do Loop limitada ao caminho crítico



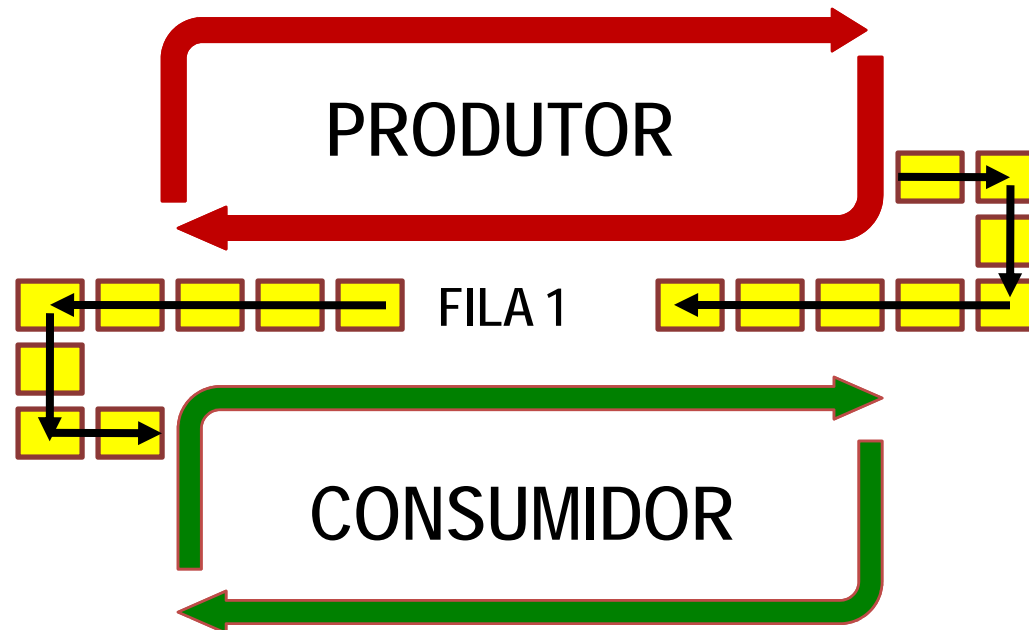
Padrões com Múltiplos Loops

- Loops separados podem executar em paralelo
- Permite aplicações com múltiplas velocidades



Padrão de Projeto Produtor Consumidor

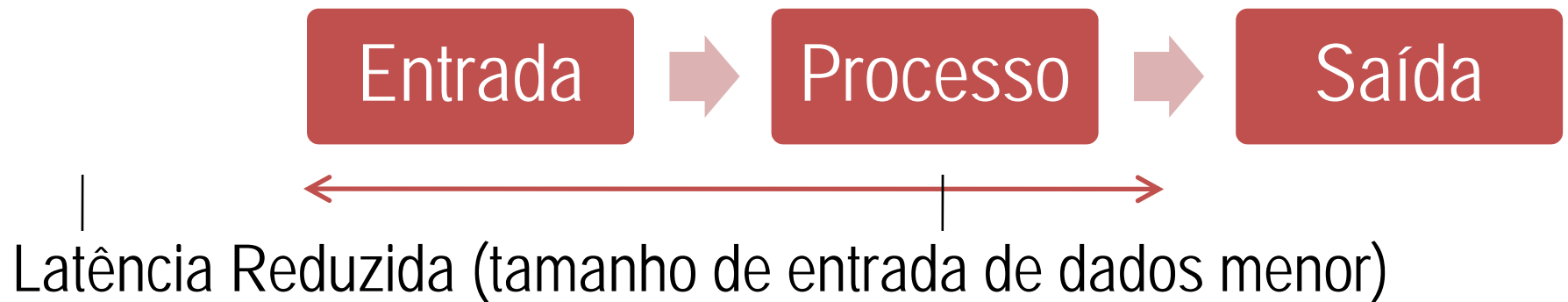
- Queues podem armazenar elementos quando transferência de dados é necessária entre loops
- O padrão produtor/consumidor é um dos padrões mais comumente utilizados



Estudo de Caso: Minimizando Latência de Entrada/Saída



Latência de Entrada/Saída



Estudo de Caso: Processamento Sustentável

Processamento
de Sinal



A medida que a taxa de processamento de sinal é maior que a taxa de entrada, a aplicação é **sustentável**

Estudo de Caso: Processamento Sustentável (continuação)

Processamento
de Sinal



Se a taxa de processamento de sinal é menor que a taxa de entrada, a aplicação é **não sustentável!**

Estudo de Caso: Sumário

- Realizar processamento de sinal on-the-fly
- Mínimo atraso entre entrada/saída (latência)
 - Atinja isso com conjuntos de dados menores
- Necessário certificar-se que o processamento de sinal ocorra pelo menos tão rápido quanto a entrada é recebida

Meta: Melhore uma aplicação de entrada/saída usando técnicas de programação paralela

Exercício:

Estudo de Caso Produtor/Consumidor

LabVIEW para Programação Multicore

Linguagem de Programação

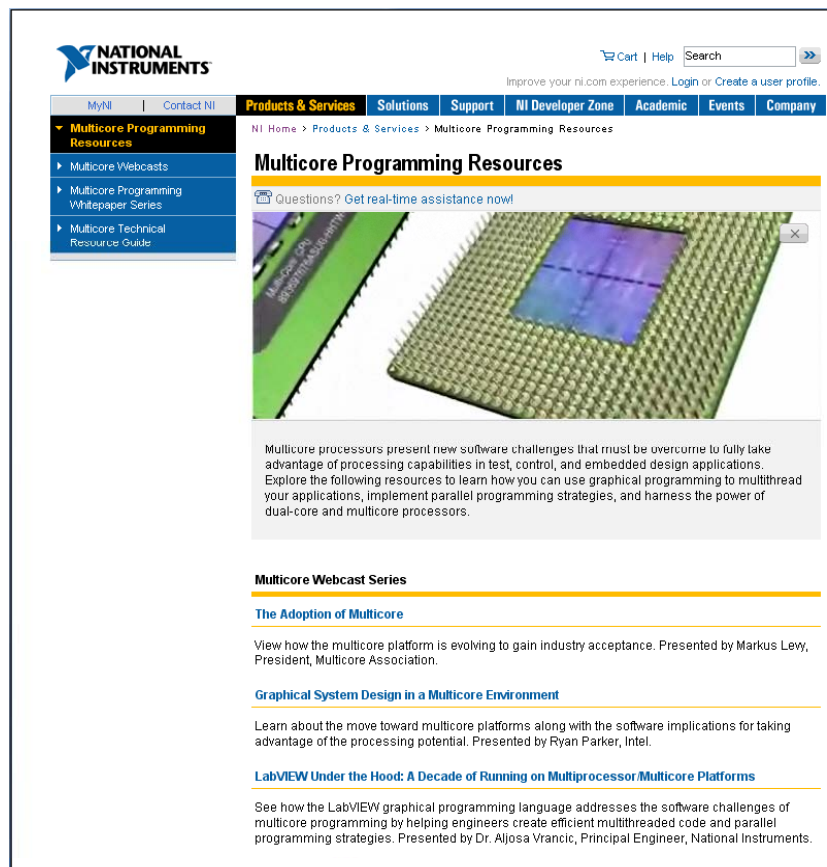
1. Programação gráfica permite aos desenvolvedores “ver” o paralelismo no código fonte
2. O paradigma do fluxo de dados **garante** sincronismo
3. Multithreading automático

Tecnologias Chave

1. Estruturas para permitir controle de baixo nível de thread
2. Ferramentas de depuração para verificação funcional e rastreamento de erros
3. Uso de bibliotecas otimizadas da Intel (Biblioteca Principal Matemática - MKL)

Recursos Adicionais

www.ni.com/multicore



The screenshot shows the National Instruments website interface. At the top, there's a navigation bar with links like 'MyNI', 'Contact NI', 'Products & Services', 'Solutions', 'Support', 'NI Developer Zone', 'Academic', 'Events', and 'Company'. A search bar is also present. Below the navigation bar, a sidebar on the left lists 'Multicore Programming Resources' with sub-links for 'Multicore Webcasts', 'Multicore Programming Whitepaper Series', and 'Multicore Technical Resource Guide'. The main content area is titled 'Multicore Programming Resources' and features a large image of a multicore processor. Below the image, there's a paragraph explaining the challenges of multicore programming and the need for graphical programming. Further down, there's a section titled 'Multicore Webcast Series' with three sub-sections: 'The Adoption of Multicore', 'Graphical System Design in a Multicore Environment', and 'LabVIEW Under the Hood: A Decade of Running on Multiprocessor Multicore Platforms'. Each sub-section has a brief description of the content.

NATIONAL INSTRUMENTS

Cart | Help Search

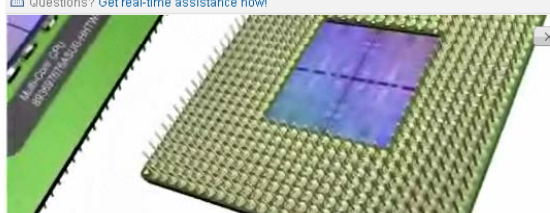
Improve your ni.com experience. [Login](#) or [Create a user profile](#).

[MyNI](#) | [Contact NI](#) | [Products & Services](#) | [Solutions](#) | [Support](#) | [NI Developer Zone](#) | [Academic](#) | [Events](#) | [Company](#)

[NI Home](#) > [Products & Services](#) > [Multicore Programming Resources](#)

Multicore Programming Resources

Questions? [Get real-time assistance now!](#)



Multicore processors present new software challenges that must be overcome to fully take advantage of processing capabilities in test, control, and embedded design applications. Explore the following resources to learn how you can use graphical programming to multithread your applications, implement parallel programming strategies, and harness the power of dual-core and multicore processors.

Multicore Webcast Series

[The Adoption of Multicore](#)

View how the multicore platform is evolving to gain industry acceptance. Presented by Markus Levy, President, Multicore Association.

[Graphical System Design in a Multicore Environment](#)

Learn about the move toward multicore platforms along with the software implications for taking advantage of the processing potential. Presented by Ryan Parker, Intel.

[LabVIEW Under the Hood: A Decade of Running on Multiprocessor Multicore Platforms](#)

See how the LabVIEW graphical programming language addresses the software challenges of multicore programming by helping engineers create efficient multithreaded code and parallel programming strategies. Presented by Dr. Aljosa Vrancic, Principal Engineer, National Instruments.



Obrigado!

Não esqueça de preencher a avaliação.

Para mais informações acesse ni.com ou
ligue para (11) 3149-3149

