

# Programación en LabVIEW para Ambientes Multinúcleo

# Agenda

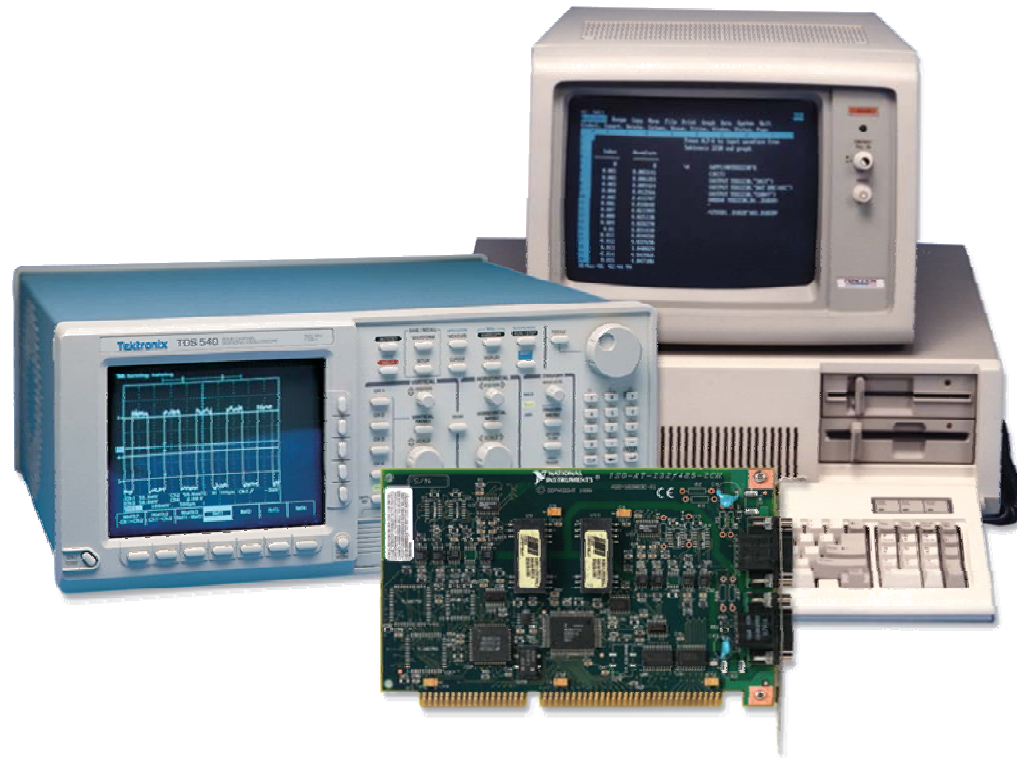
- Introducción al *Multithreading* en LabVIEW
- Técnicas de Programación en Paralelo
- Consideraciones de Tiempo Real
- Recursos

# Evolución de la Instrumentación



**Instrumentación tradicional**

# Evolución de la Instrumentación

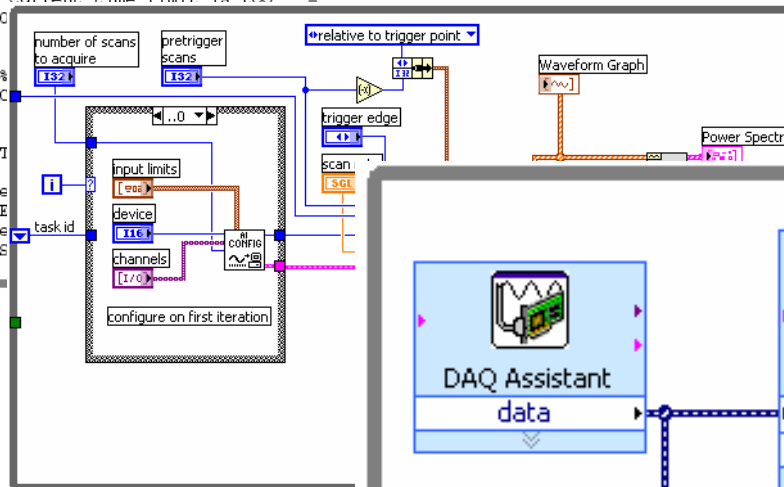


**Nace la Instrumentación Virtual**

# Evolución en la Programación

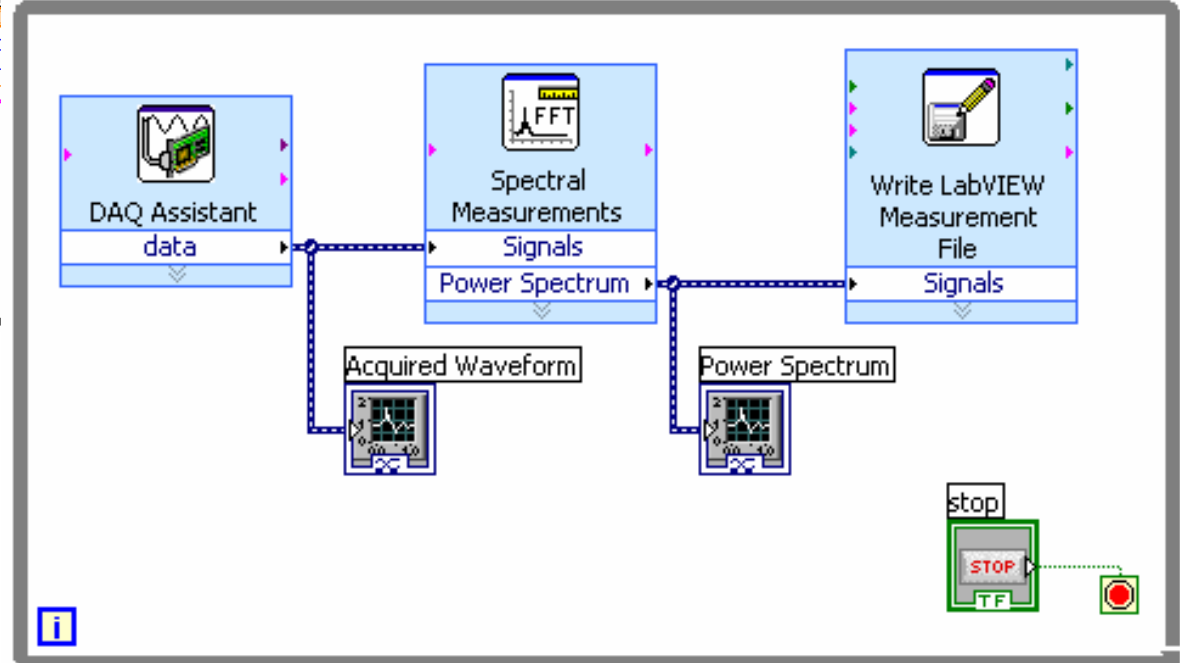
```
455 REM
460 CMD$ = CHR$(&H8) : CALL IBCMD (BRD0$,CMD$)
470 IF IBSTA% < 0 THEN GOTO 3000
480 REM
490 REM Wait for the DVM to set SRQ or for a
500 REM timeout; if the current time limit is too
510 REM short, use IBTMO
515 REM
520 MASK% = &H5000
530 CALL IBWAIT (BRD0$
540 IF (IBSTA% AND &HC
550 REM
560 REM Since neither a
570 REM occurred, IBWAIT
580 REM SRQ. Next do a
590 REM unaddress bus de
600 REM Poll Enable (SPE
610 REM DVM's talk addre
615 REM address &H20 (AS
620 REM
```

*Basada en texto*

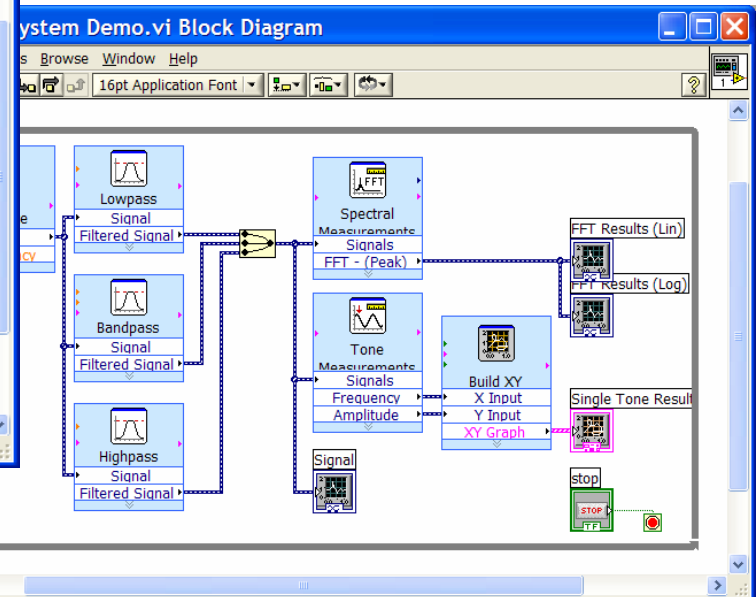
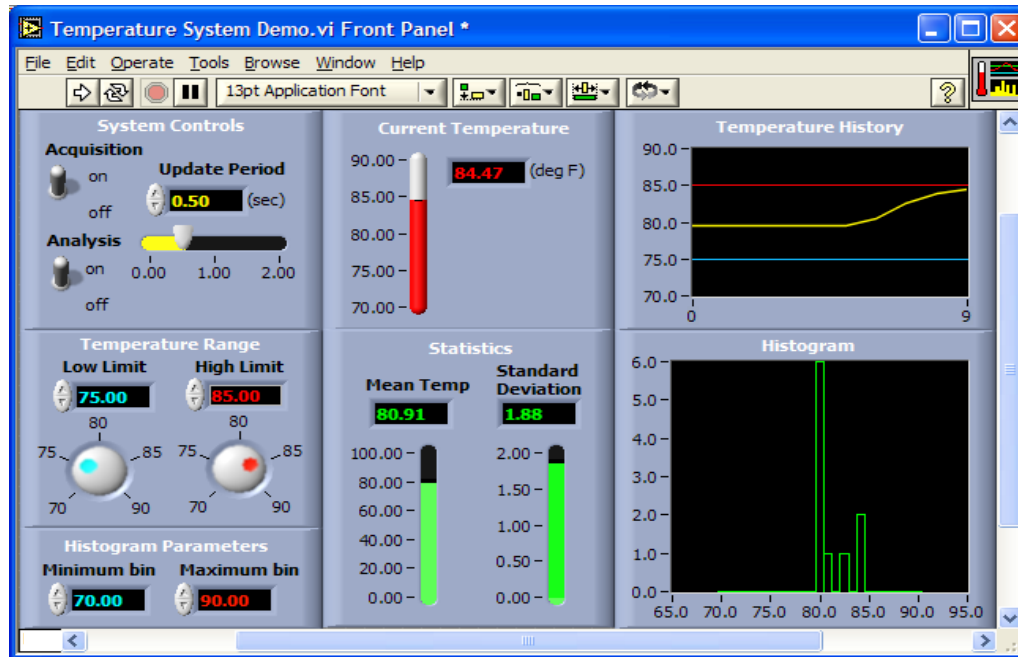


*Desarrollo gráfico*

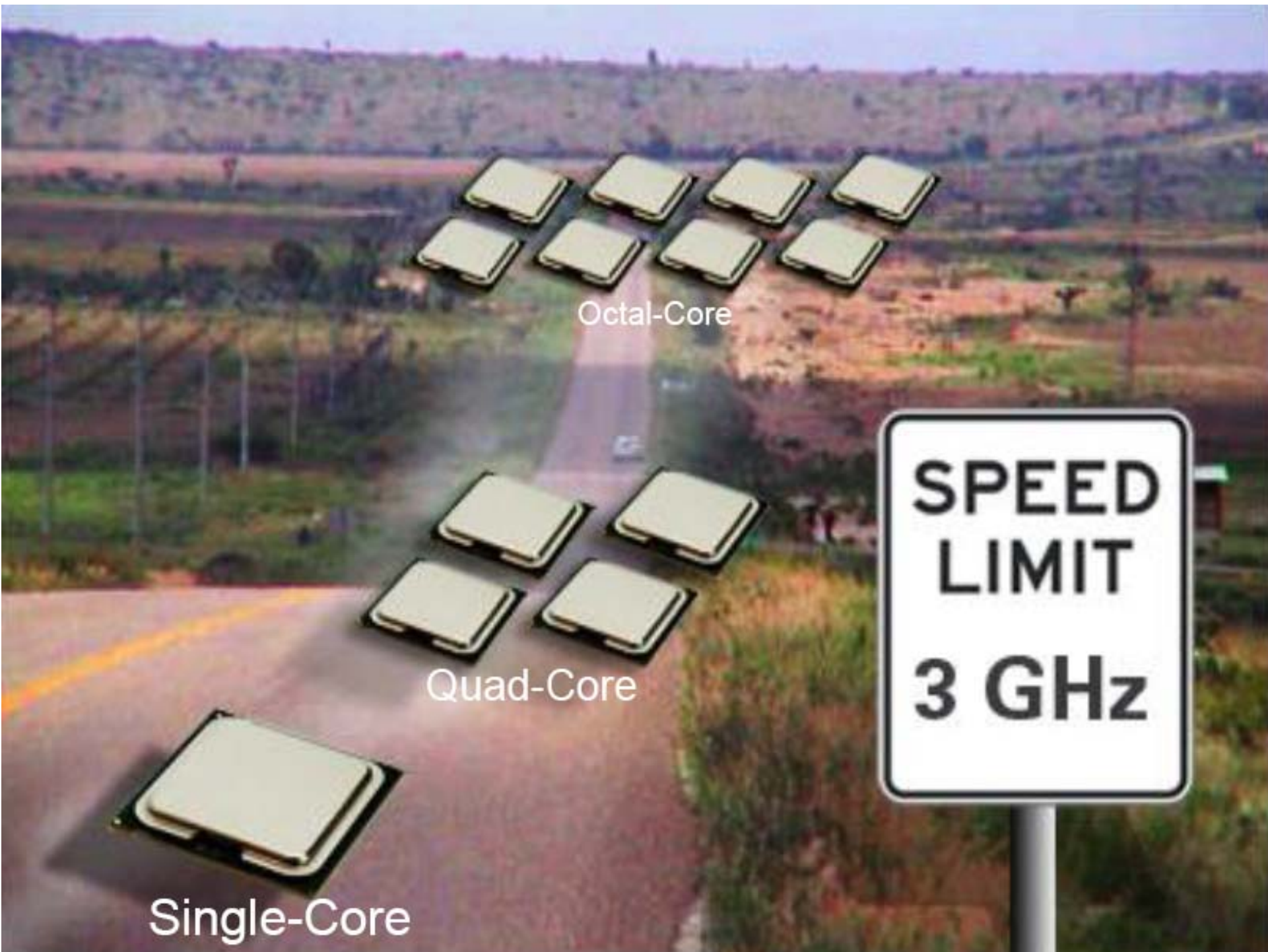
*Tecnología Express*



# Ambiente de Desarrollo Gráfico

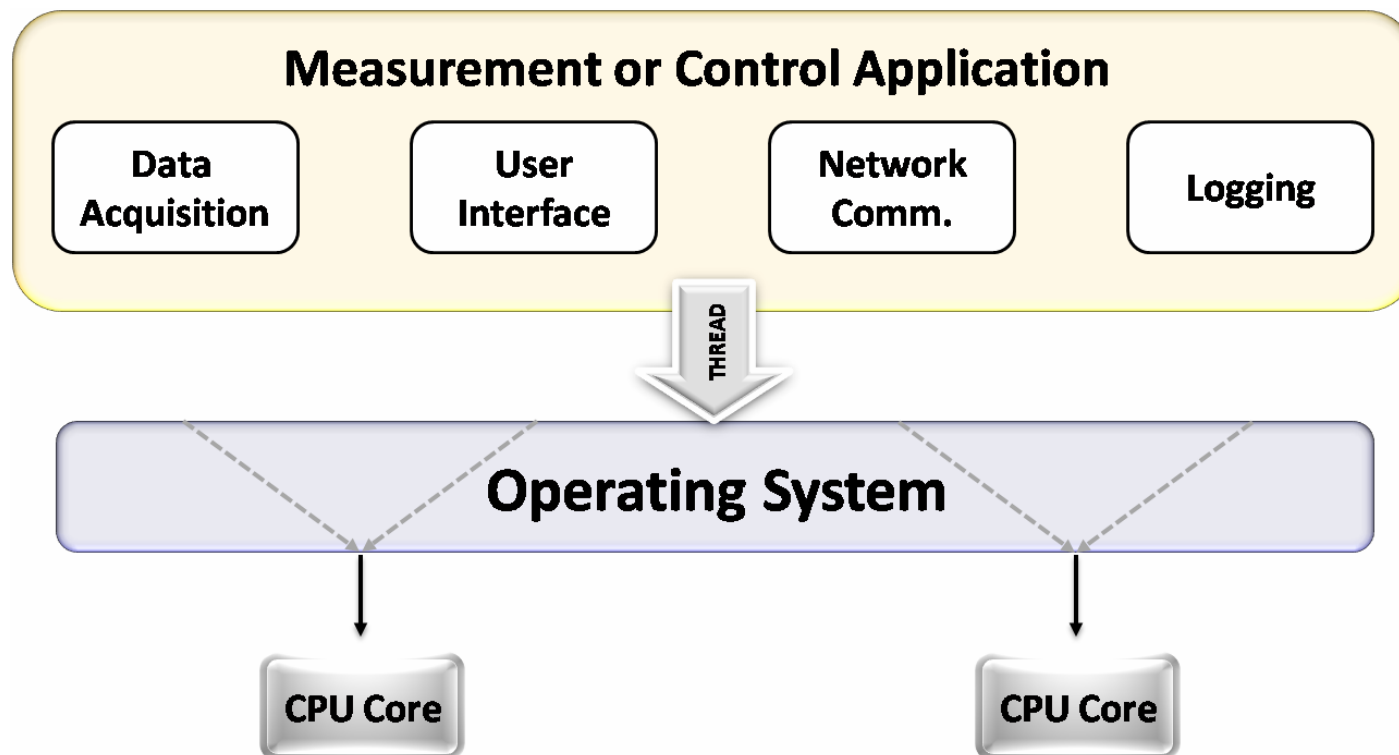






# Impacto en Ingenieros e Investigadores

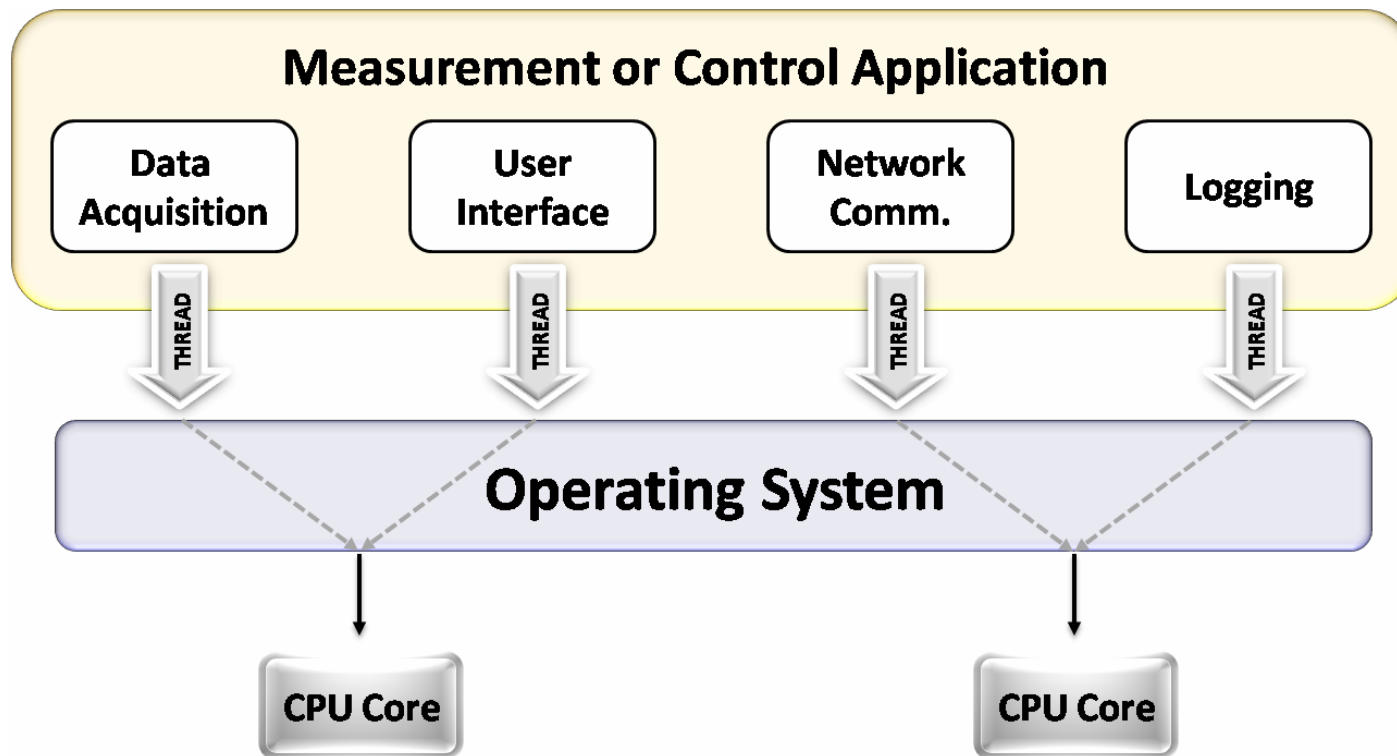
Las aplicaciones de investigadores e ingenieros se encuentran típicamente en sistemas dedicados.





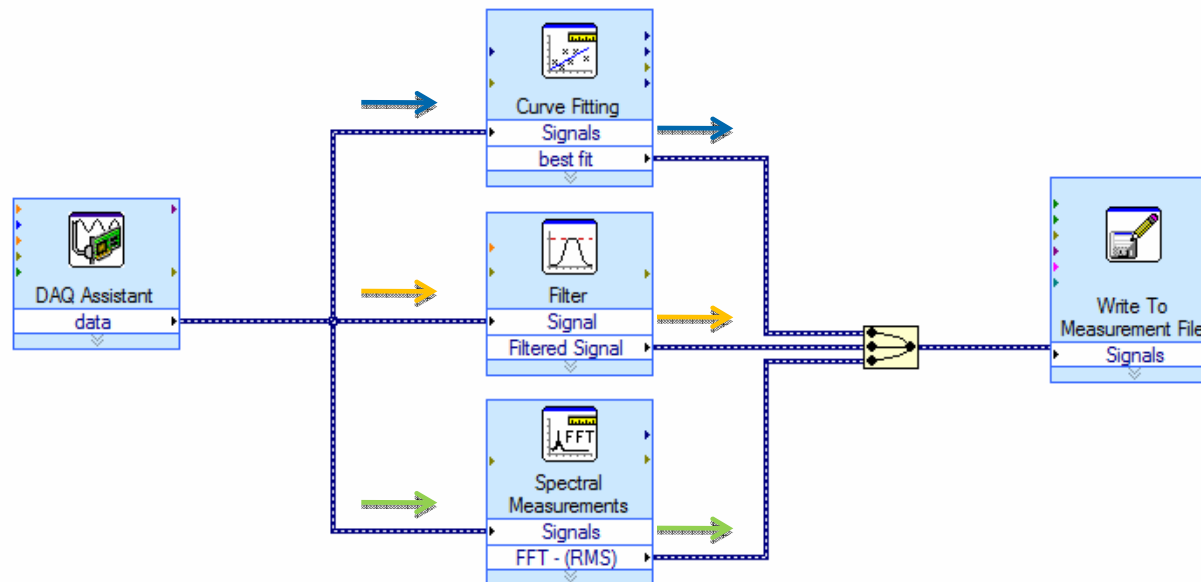
# Creando Aplicaciones Multihilo

Investigadores e ingenieros **deben** utilizar hilos (o *threads*) para beneficiarse de procesadores multinúcleo.



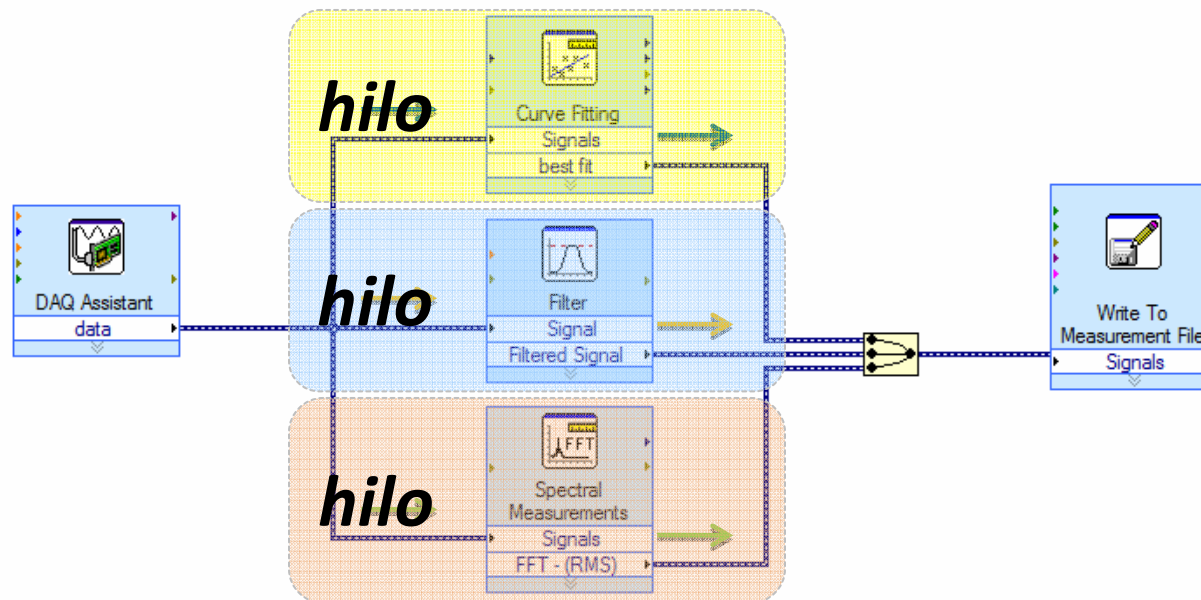
# Multihilo Automático en LabVIEW

- LabVIEW divide automáticamente cada aplicación en varios hilos de ejecución
- LabVIEW introdujo multihilo en 1998



# Multihilo Automático en LabVIEW

- LabVIEW divide automáticamente cada aplicación en varios hilos de ejecución
- LabVIEW introdujo multihilo en 1998



# Demostración: “Paralelismo Accidental” en LabVIEW

# Nuevas Funciones Multihilo en LabVIEW 8.5

- Adaptabilidad del número de hilos de ejecución del sistema dependiendo de los núcleos disponibles
- Calendarización de hilos mejorada para ciclos temporizados de LabVIEW
- Asignar un procesador específico a estructuras temporizadas
- Funciones de tiempo real:
  - Soporte para objetivos de tiempo real con multiprocesamiento simétrico
  - Real-Time Execution Trace Toolkit 2.0



# Caso de Estudio – Eaton Corporation

Eaton desarrolló un sistema de pruebas en vehículo portátil para transmisiones automotrices utilizando LabVIEW.

- Adquisición y análisis de 16 canales con un núcleo utilizando el driver multihilo NI-DAQmx
- Ahora adquieren y analizan más de 80 canales en multinúcleo

*“No hubo necesidad de reescribir nuestra aplicación para las nuevas plataformas de procesamiento multinúcleo.”*

Scott Sirrine  
Ingeniero Líder en Diseño  
Eaton División Camiones



# Soporte en Software para Sistemas Multihilo

## Stack de Software

<b>Herramienta de Desarrollo</b>	Soporte proporcionado en el sistema operativo de elección; la herramienta facilita la creación de hilos y optimización	✓ Ejemplo: Naturaleza multihilo de LabVIEW y estructuras que proporcionan optimización
<b>Librerías</b>	Capacidad de trabajar en ambientes multihilo, librerías reentrantes	✓ Ejemplo: Librerías BLAS
<b>Controladores de dispositivos</b>	Controladores diseñados para desempeño óptimo multihilo.	✓ Ejemplo: Controlador NI-DAQmx
<b>Sistema Operativo</b>	Sistema operativo soporta multihilo y multitarea y puede balancear la carga de tareas	✓ Ejemplo: Soporte para Windows, Mac OS, Linux® OS, y sistemas operativos de tiempo real



# ¿Ejecutará Más Rápida mi Aplicación de LabVIEW en un Sistema Multinúcleo?

## Consideraciones Claves

- ¿El código es secuencial o paralelo?
- ¿Cómo son utilizados los recursos compartidos en la aplicación?

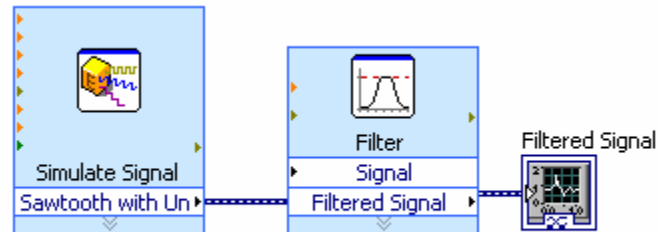
## Conclusiones

- La mejora en velocidad de ejecución depende de la aplicación
- Muchas aplicaciones requerirán modificaciones pequeñas para aprovechar al máximo el procesador multinúcleo

# Código Secuencial contra Código Paralelo

## *Ejemplo Secuencial*

- LabVIEW ejecutara VIs en orden, uno después de otro.



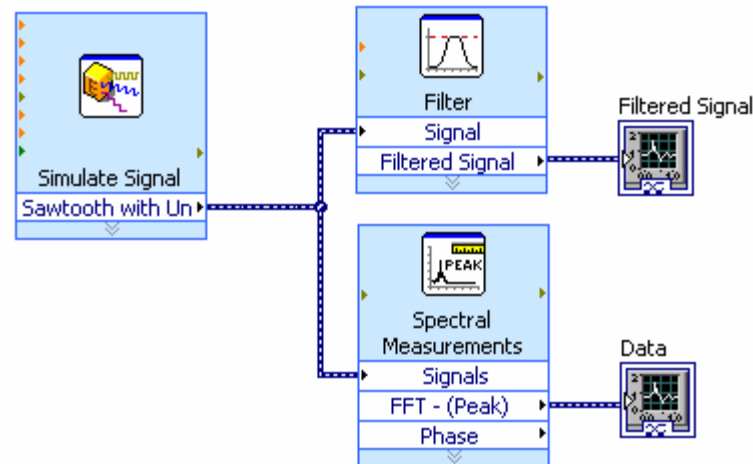
## **Resultado:**

Un sistema multinúcleo no mejorará la velocidad de procesamiento sobre un sistema de un solo núcleo porque no existe paralelismo en el código

# Código Secuencial contra Código Paralelo

## *Ejemplo Paralelo*

- Debido a que el código tiene ramas paralelas, LabVIEW tratará de compilar el código para su ejecución paralela



## RESULTADO:

El código puede aprovechar los beneficios de sistemas multinúcleo, por ejemplo las mediciones de espectro y filtrado se ejecutarán en paralelo

# Potenciales Cuellos de Botella:

## Recursos Compartidos

### 1. Dependencia de Datos

**Ejemplo:** Datos almacenados en variables globales que necesitan ser utilizadas por diferentes VIs serían un recurso compartido

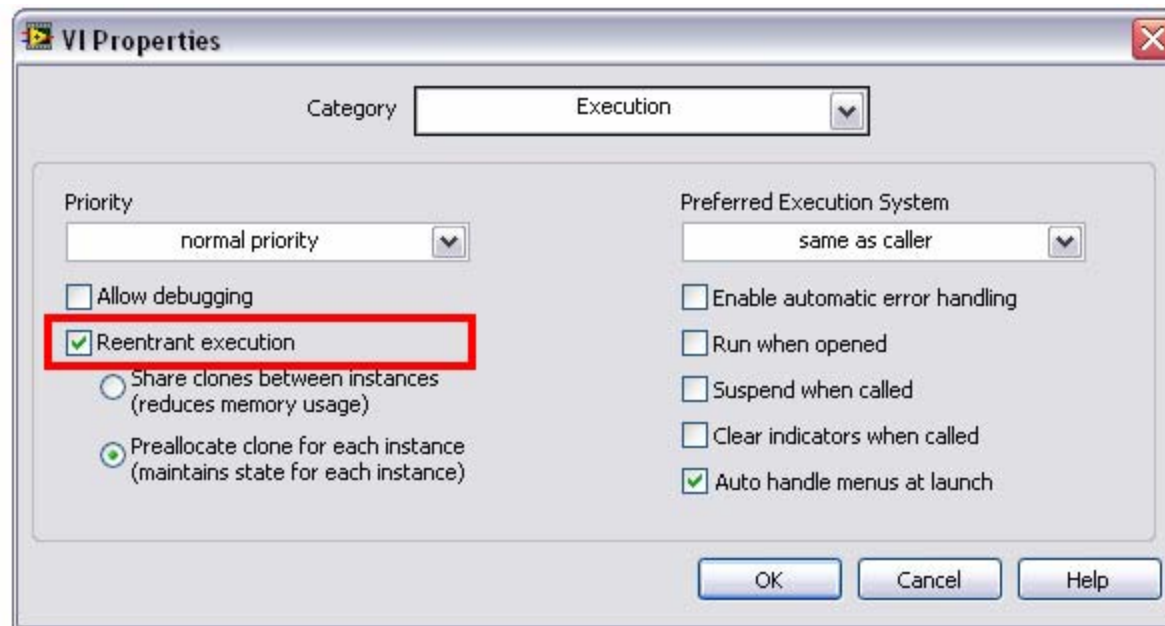
### 2. Disco Duro

**Ejemplo:** Las computadoras pueden leer o escribir a disco duro un elemento a la vez (la entrada/salida de archivo no se puede realizar en paralelo)

### 3. VIs no-reentrant

# VIs No-Reentrantes

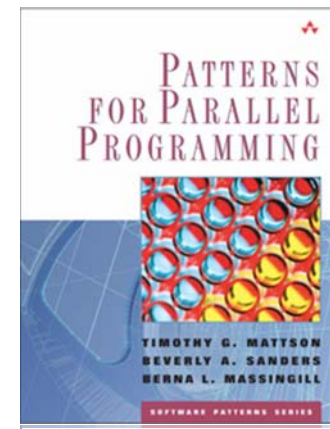
- VIs que son no-reentrantes no se pueden llamar simultáneamente; una llamada se ejecuta y el otro espera a que el primero termine antes de ejecutarse.



Para hacer un VI reentrante, seleccionar **File»VI Properties**, seleccionar la categoría de **Execution** , y seleccionar **Reentrant execution**.

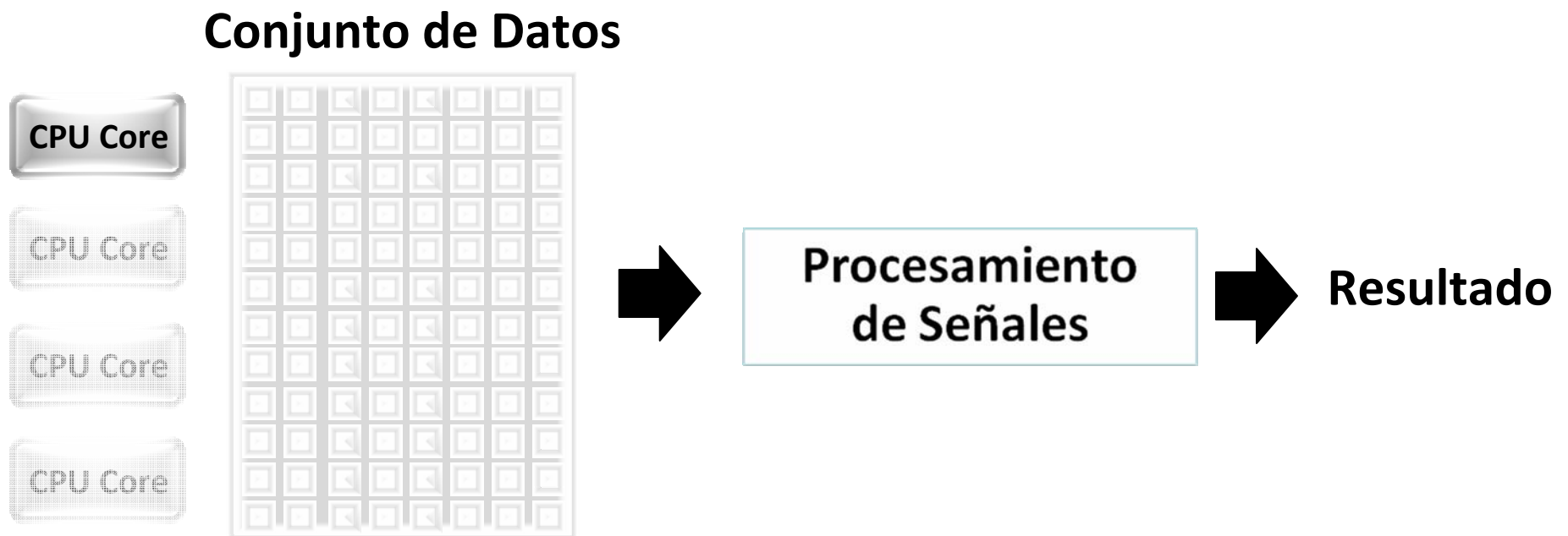
# Técnicas de Programación Paralela para Mejorar el Desempeño en Sistemas Multinúcleo

- Paralelismo de Tareas
- Paralelismo de Datos
- “Pipelining”



# Paralelismo de Datos

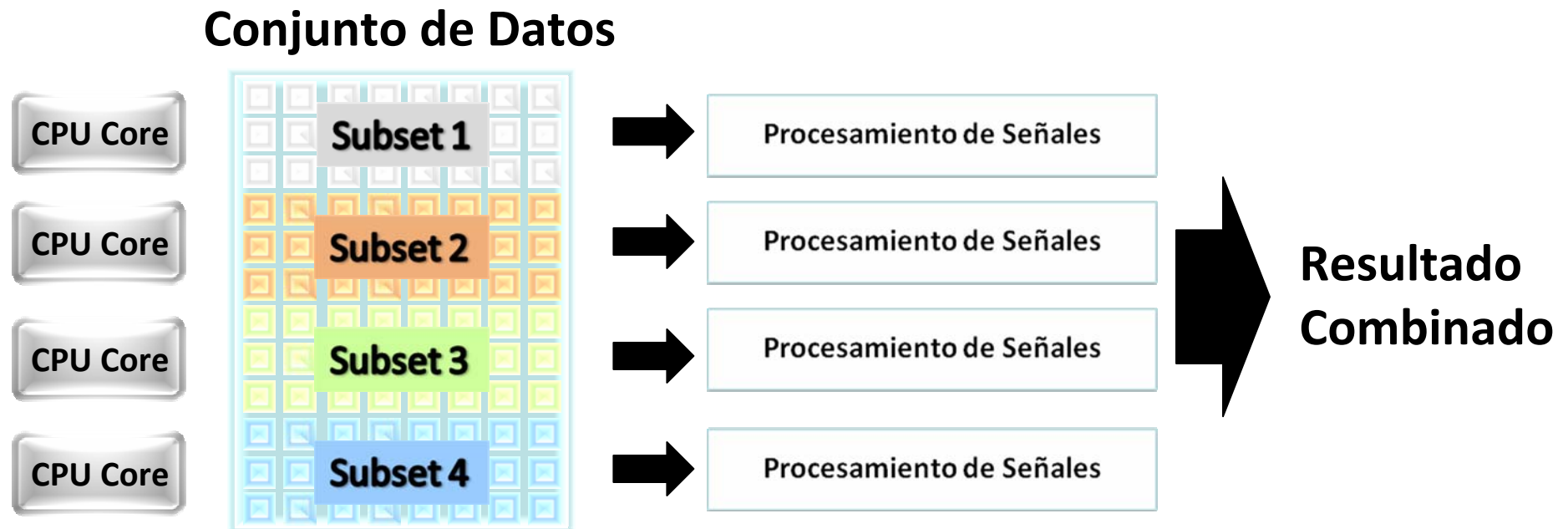
Se pueden acelerar las operaciones intensivas en procesamiento sobre conjuntos grandes de datos en sistemas multinúcleo





# Paralelismo de Datos

Se pueden acelerar las operaciones intensivas en procesamiento sobre conjuntos grandes de datos en sistemas multinúcleo



# Demostración de Paralelismo de Datos

# Ejemplo de Aplicación: Control de Alta Velocidad

- Instituto Max Planck (Munich, Alemania)
- Control de plasma en fusión nuclear tokamak con LabVIEW en sistema de ocho núcleos utilizando la técnica de paralelismo de datos.

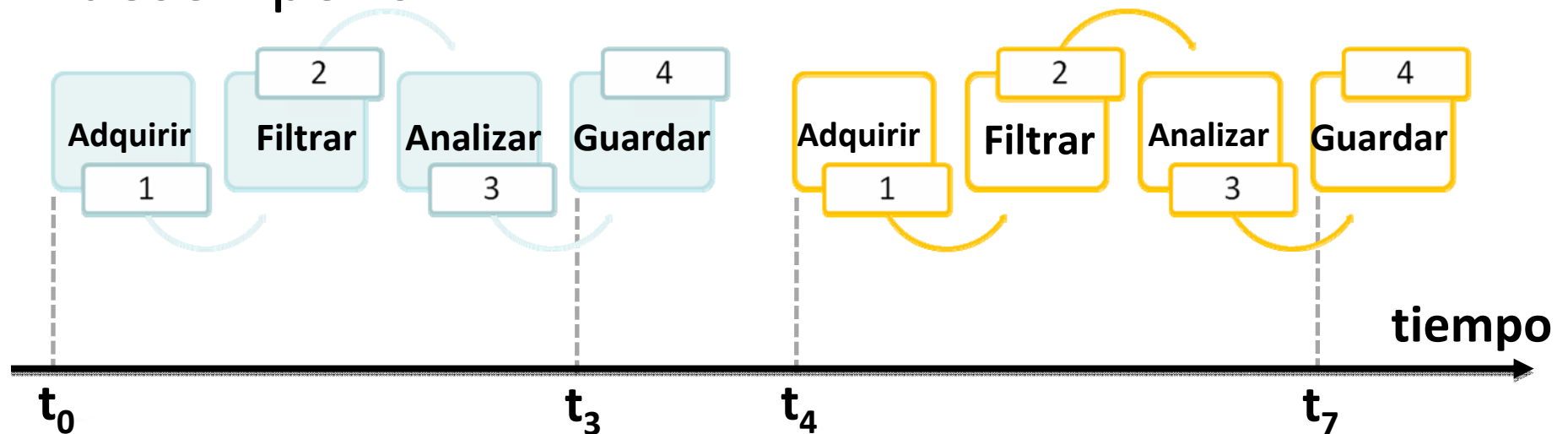
*“...con LabVIEW, obtuvimos un aumento en procesamiento de 20 veces mas con una máquina de ocho núcleos sobre una de un solo núcleo...”*

Louis Giannone

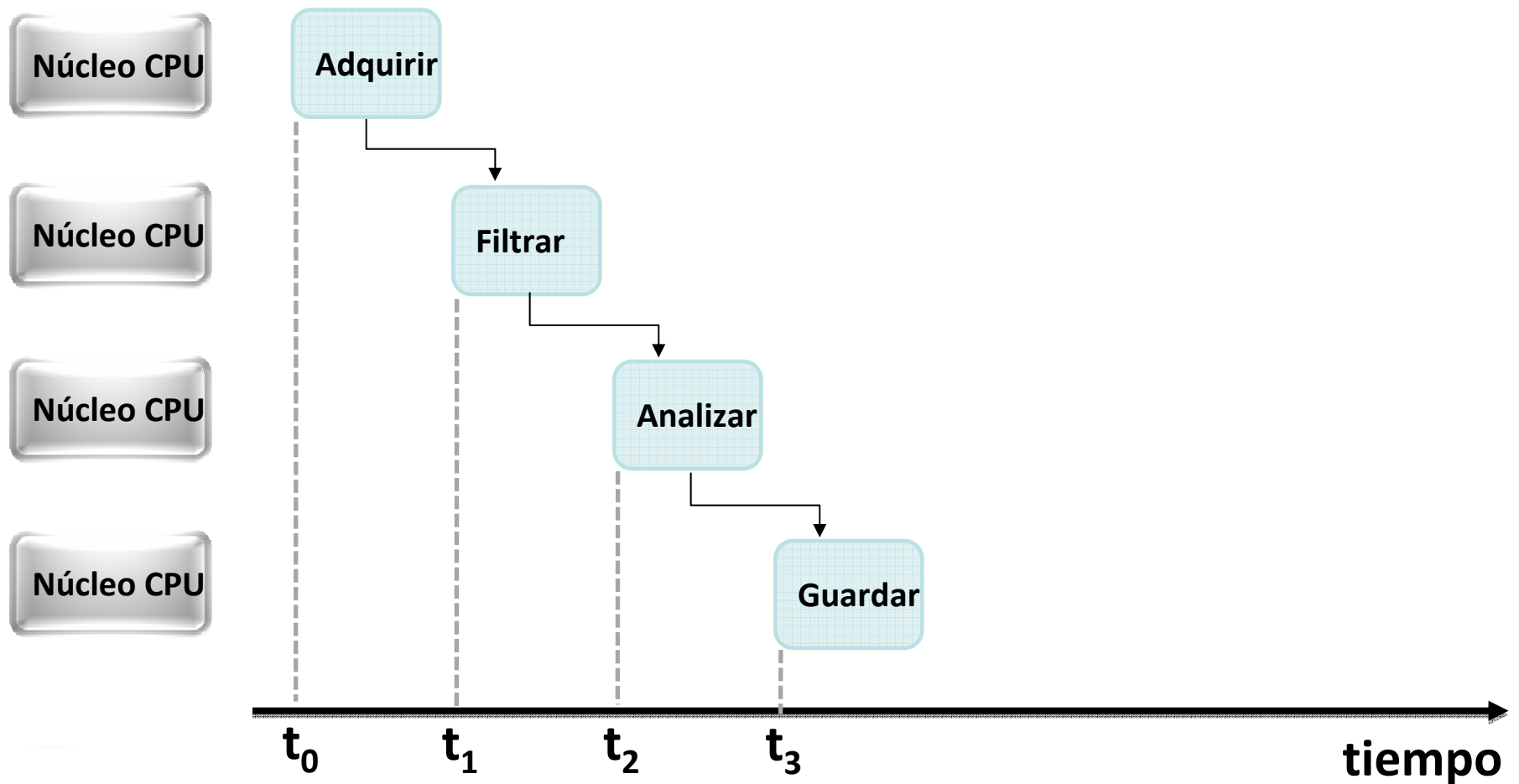


# Estrategia de “Pipelining”

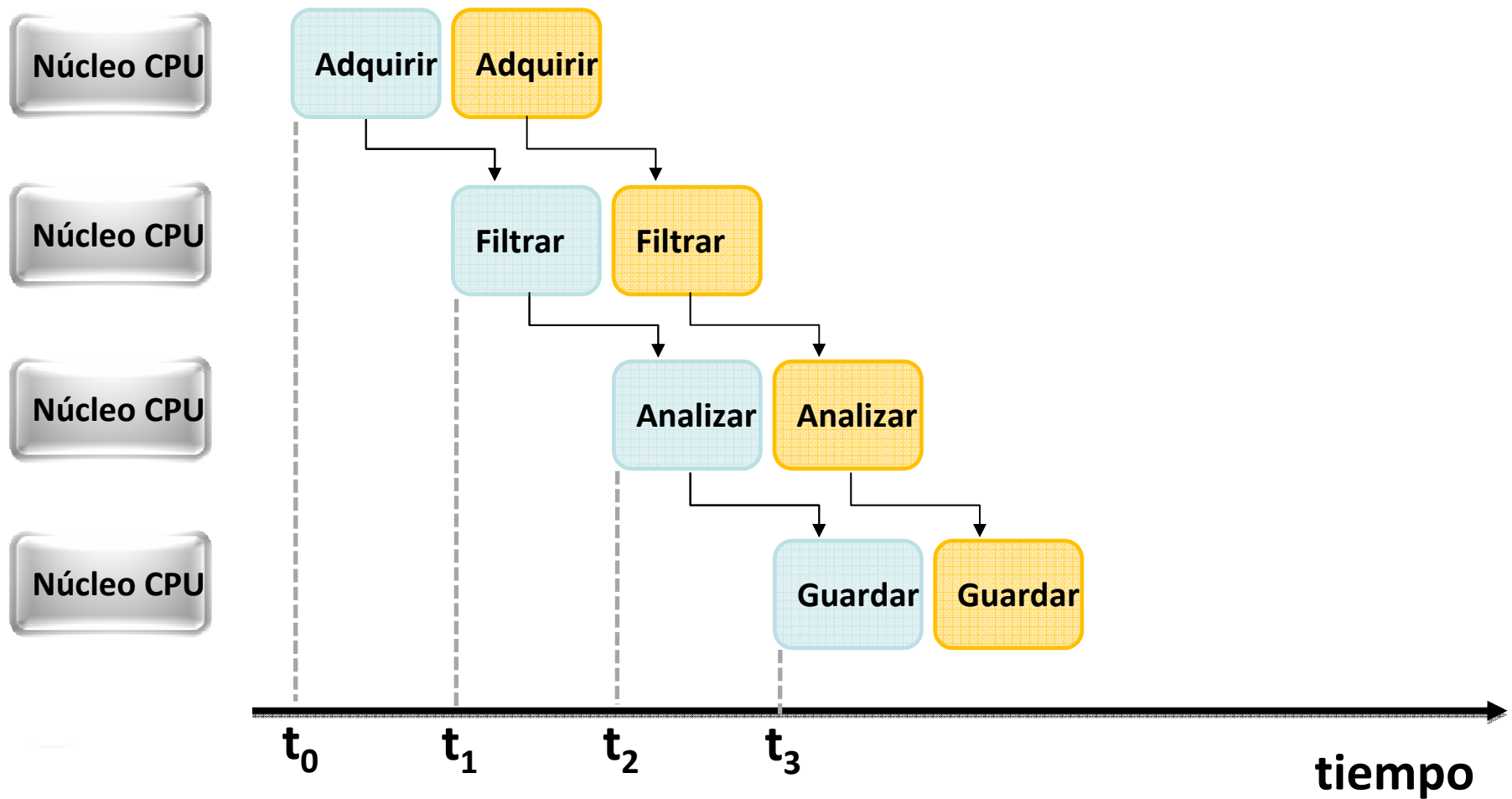
- Muchas aplicaciones involucran algoritmos multipasos secuenciales
- Aplicar “pipelining” puede aumentar el desempeño



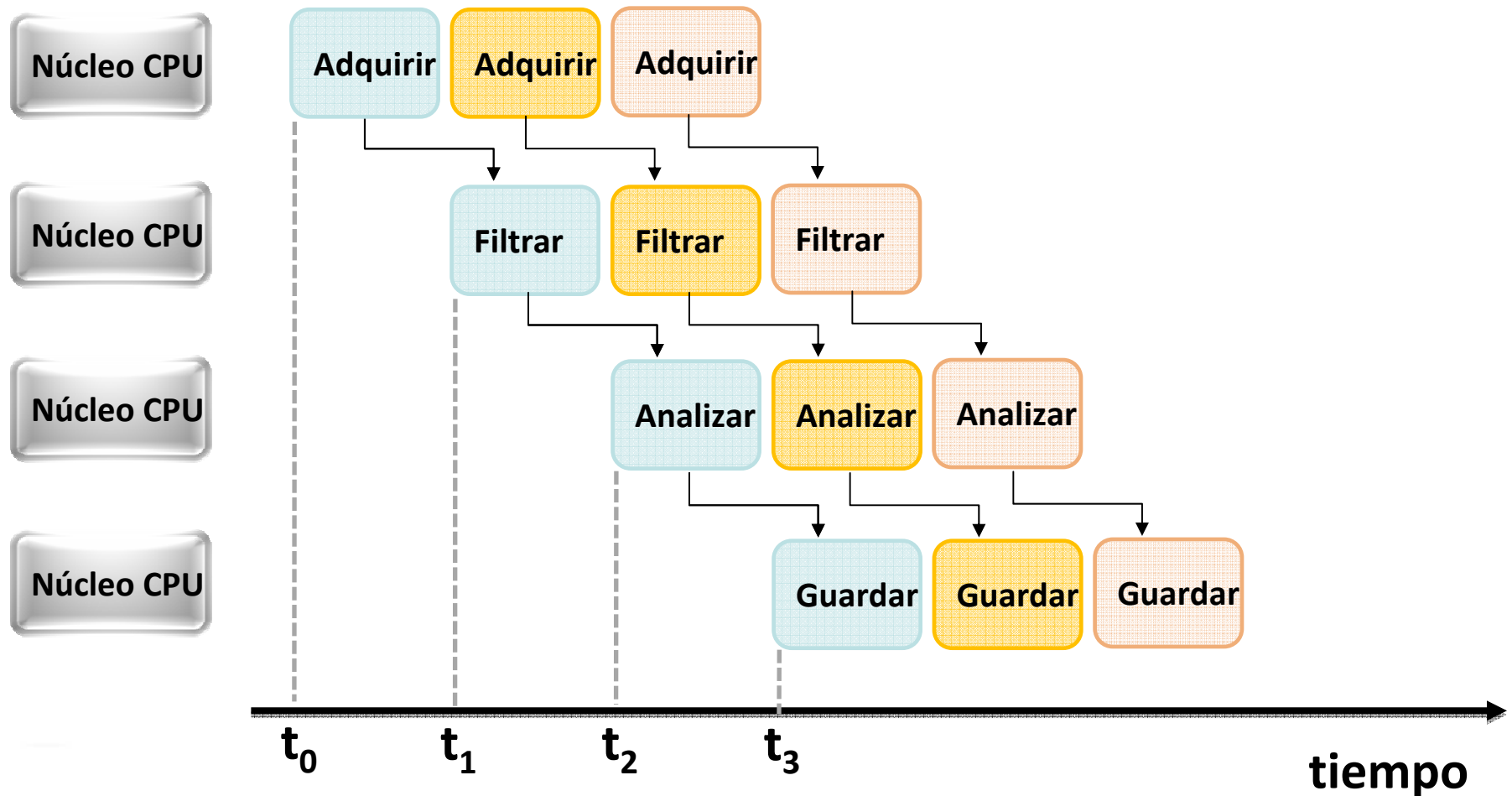
# Estrategia de “Pipelining”



# Estrategia de “Pipelining”



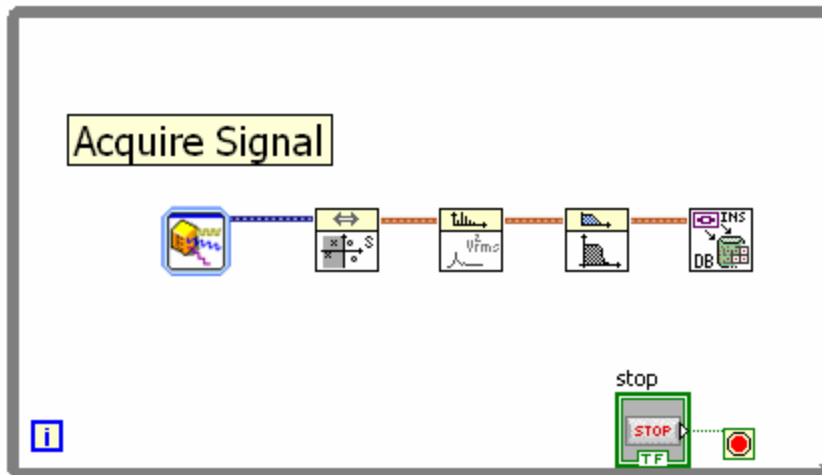
# Estrategia de “Pipelining”



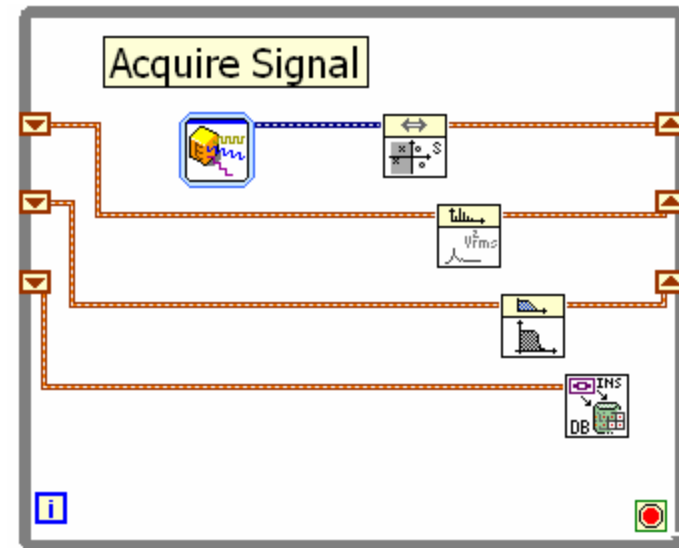


# “Pipelining” en LabVIEW

## Secuencial

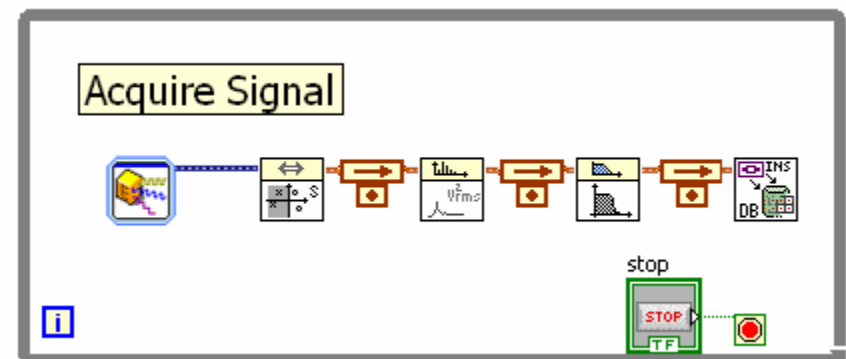


## “Pipelined”



0

Nota: Los búfers también se pueden utilizar para “apilar” datos entre diferentes ciclos



# Demostración de “Pipelining”

# Consideraciones Claves para “Pipelining”

- Considerar el número de núcleos en el procesador para determinar el número de etapas de “pipeline”
- Asegurarse de balancear las etapas, porque la etapa más larga limitará el aumento de desempeño

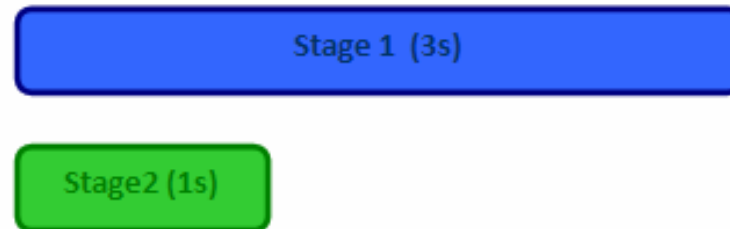
– Ejemplo:

“Pipeline”  
desbalanceado

Non-Pipelined (total time = 4s)

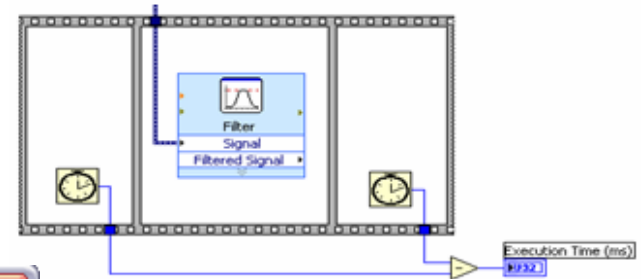


Pipelined (total time = 3s): Speed-up = 1.33X (not an ideal case for pipelining)



# Recomendaciones para Balancear Etapas de “Pipeline”

- Utilice técnicas de comparación de LabVIEW
  - Realice comparaciones básicas estampas de tiempo y el VI Profiler.



Profile Performance and Memory - Multicore Demos.lvproj

☒ Timing statistics    ☐ Profile memory usage  
☒ Timing details    ☐ Memory usage

Time unit: milliseconds    Size unit: kilobytes    Select Application Instances...

Profile Data

	VI Time	Sub VIs Time	Total Time	# Runs	Average	Shortest
Matrix Multiply.vi	7290.5	0.0	7290.5	3	2430.2	1822.6
Create Matrix.vi	6319.1	0.0	6319.1	6	1053.2	1011.5
Data Parallelism - Matrix Multiply.vi	50.1	11486.5	11536.6	1	50.1	50.1
Random Matrices.vi	0.0	6319.1	6319.1	3	0.0	0.0

Stop    Snapshot    Save    Close    Help

## Ejemplo de Aplicación: Pruebas de Comunicaciones

- AmFax Ltd. (Reino Unido)
- Crea sistema de pruebas inalámbricas para teléfonos de la nueva generación, utilizando la técnica de “pipelining” de LabVIEW

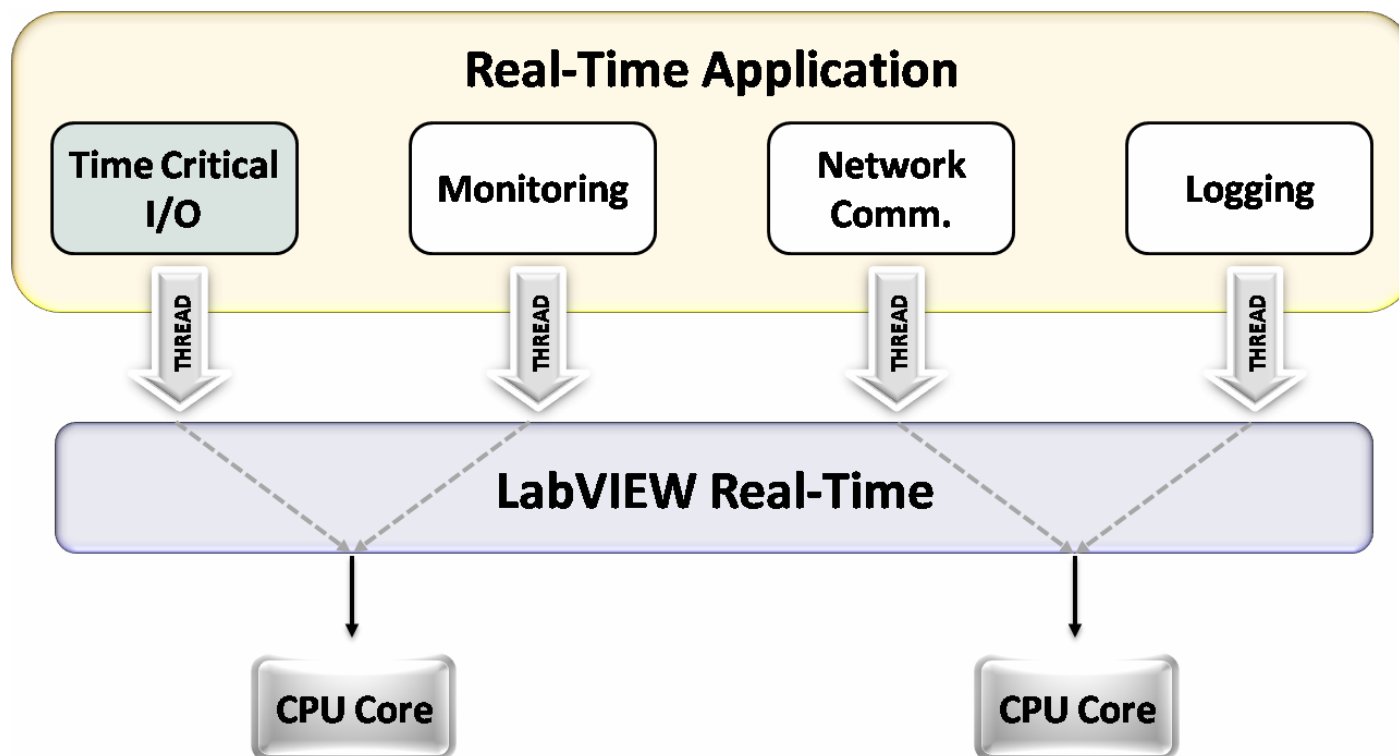
*“Con LabVIEW y el controlador embebido de doble núcleo, hemos logrado un ahorro de tiempo de 5 veces mejor que antes ... ”*

Mark Jewell  
BDM – Wireless  
AmFax Ltd.



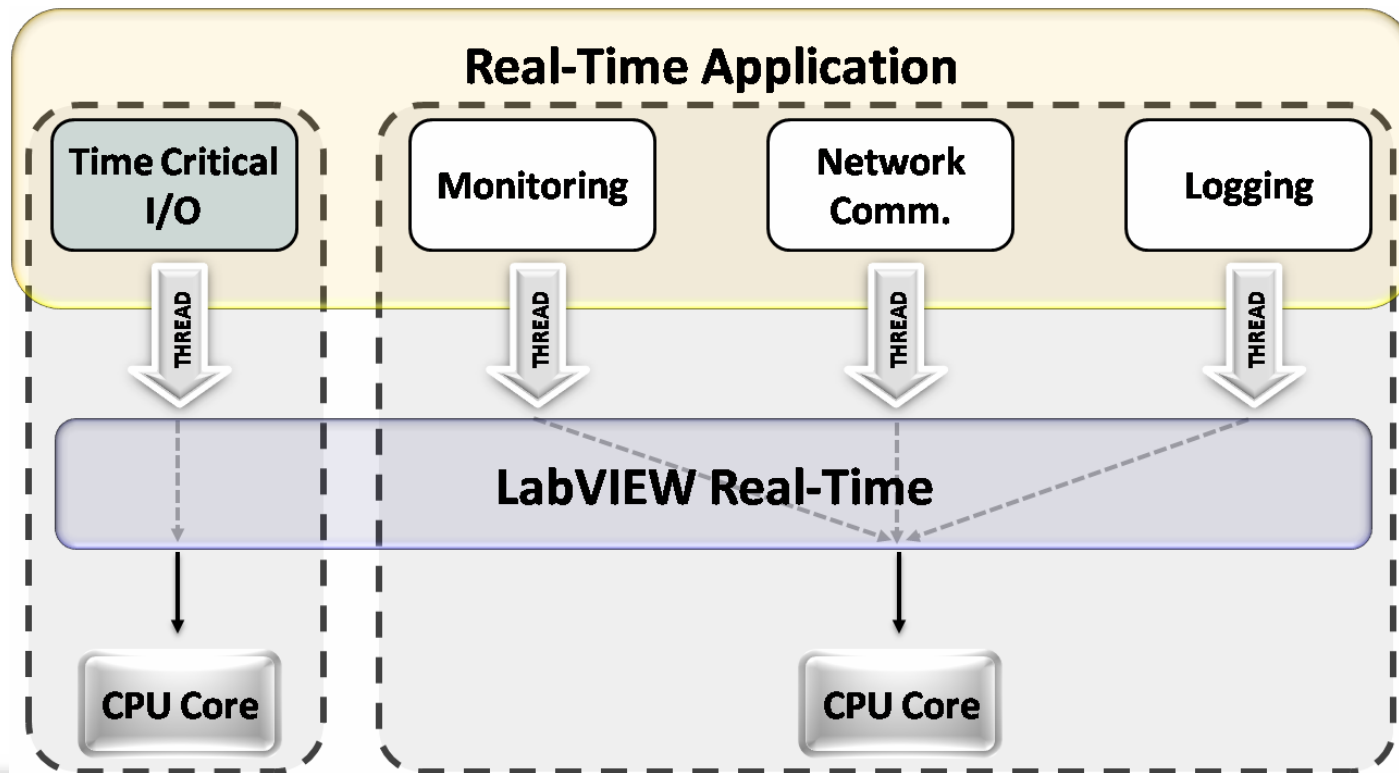
# Sistemas Determinísticos de Tiempo Real

LabVIEW 8.5 permite multiprocesamiento simétrico para sistemas de tiempo real



# Asignando Tareas a Núcleos en Específico

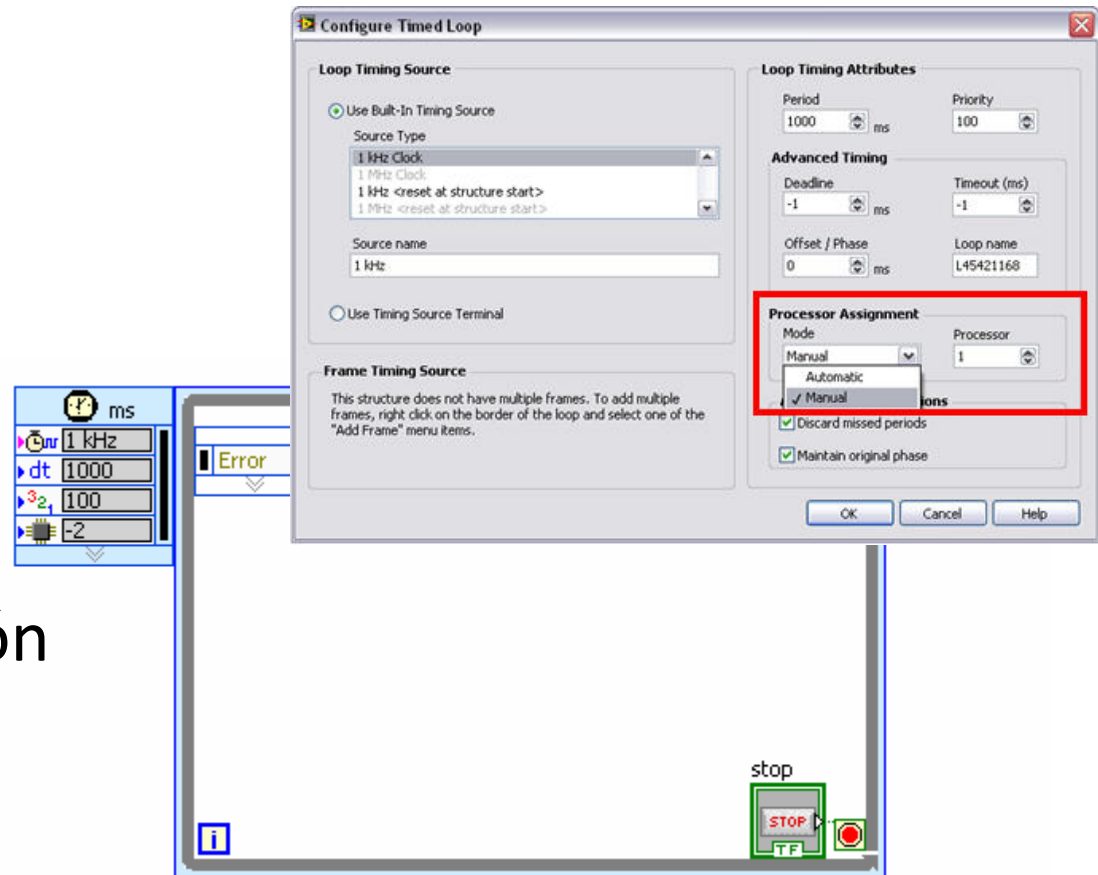
En LabVIEW 8.5, los usuarios pueden asignar código a un núcleo del procesador en específico utilizando el ciclo temporizado de LabVIEW.





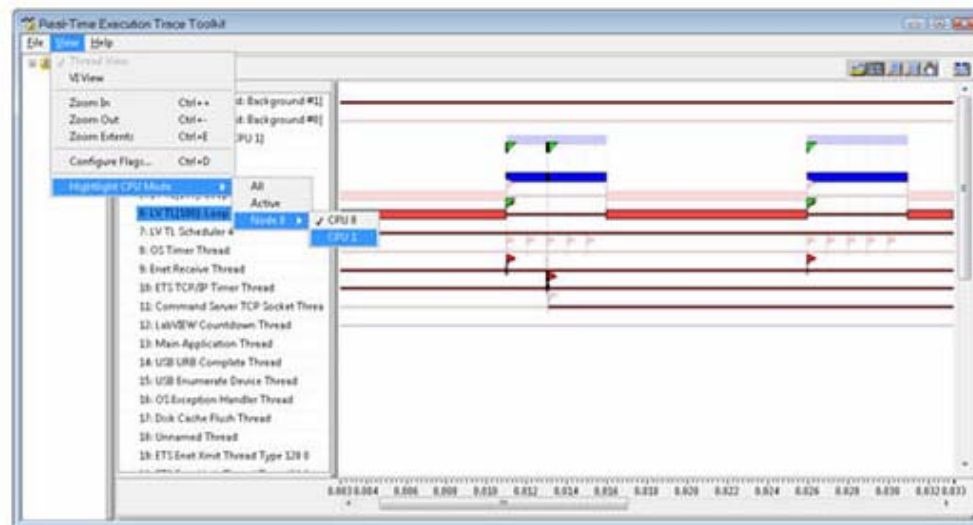
# Afinidad de Procesador con el Ciclo Temporizado

- Recomendado para desarrollo de tiempo real
- Puede utilizarse en Windows bajo uso especial (ejemplo: optimización de cache)



# Real-Time Execution Trace Toolkit 2.0

- Evaluación por 30 días
- Compatibilidad con LabVIEW 7.1 o superior
- **Nuevas funciones en la Versión 2.0**
  - Mejora en desempeño
  - Ordenamiento de actividad
  - Nuevas banderas de depuración
  - Soporte multinúcleo



# Ejemplo de Aplicación: Control en Tiempo Real

- Sistema de Túnel de Viento para Vuelos Seguros en el Centro de Investigación Ames de la NASA
- Resultados de la Comparación
  - Se ejecuto en un controlador NI PXI-8106 RT
  - El ciclo de tiempo critico fue reducido de **43% de carga en CPU a 30% de carga en CPU** en un CPU del PXI-8106 RT
  - Se dejo libre a un núcleo para procesamiento de tareas no críticas




National Aeronautics  
and Space Administration

*Fuente de Imagen <http://windtunnels.arc.nasa.gov>*

# Recursos

## [www.ni.com/multicore](http://www.ni.com/multicore)


### Multicore Programming Resources

 Questions? Get real-time assistance now!

#### Introducing LabVIEW 8.5

Get the Most from Your Multicore Processor

[View the webcast >>](#)



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 engineers and scientists can use graphical programming to implement parallel programming strategies and harness the power of multicore processors.

#### Multicore Programming Fundamentals

[View the Multicore Programming Whitepaper Series](#)

Browse the library of whitepapers to learn about multicore programming strategies, key concepts, and performance benchmarks.

[Develop High-Performance Real-Time Systems with Multicore Technology](#)

#### Technical Resources

- [Overcoming Multicore Programming Challenges with LabVIEW](#)
- [Programming Strategies for Multicore Processing: Task Parallelism](#)
- [Will my LabVIEW program run faster when I upgrade to a multicore computer?](#)