NIDays

FORO TECNOLÓGICO SOBRE
DISEÑO GRÁFICO DE SISTEMAS

Madrid 22 Marzo 2012
IFEMA - Feria de Madrid

spain.ni.com/nidays

NATIONAL INSTRUMENTS

Consejos y Trucos para mejorar el rendimiento de aplicaciones LabVIEW

Jorge Cuadrado

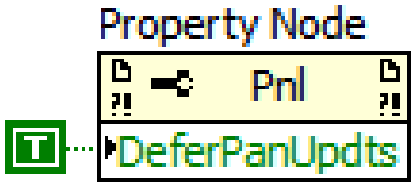
Applications Engineer

Agenda

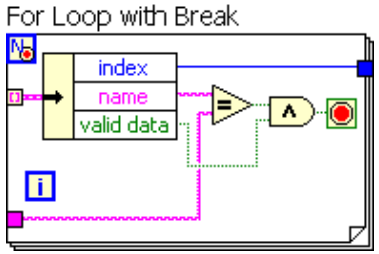
- Resumen de técnicas comunes
- Técnicas de benchmarking
- Técnicas de programación

Técnicas Comunes

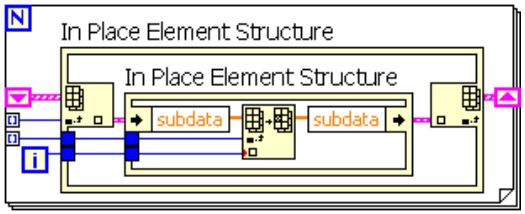
#1 – Defer Panel Updates



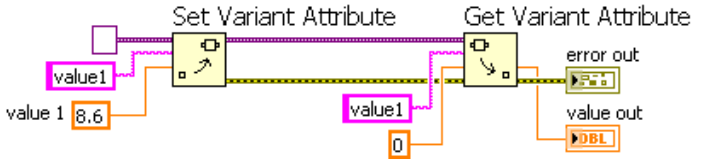
#2 – For Loop con Break



#3 – In Place Element Structure

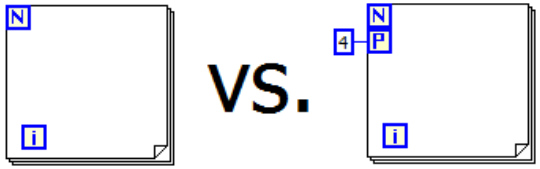


#4 – Variant Attributes



#5 – Build Array en orden

#6 – Parallel For Loop

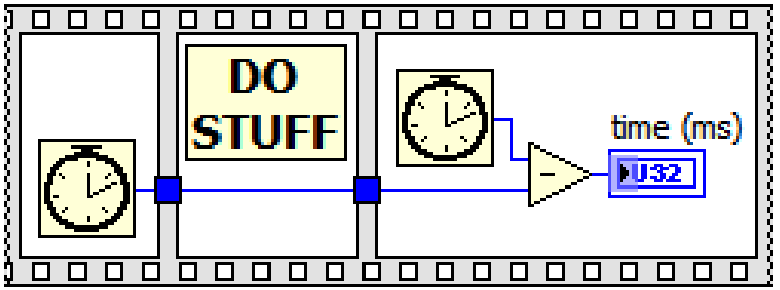


Técnicas de Benchmarking

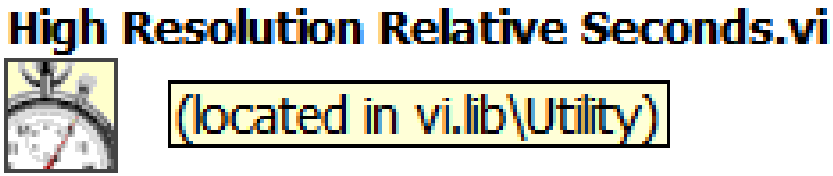
¿Cómo saber la técnica de programación que hace que nuestra aplicación funcione lo más rápido posible?

Técnicas de Benchmarking

- Bien



- Mejor



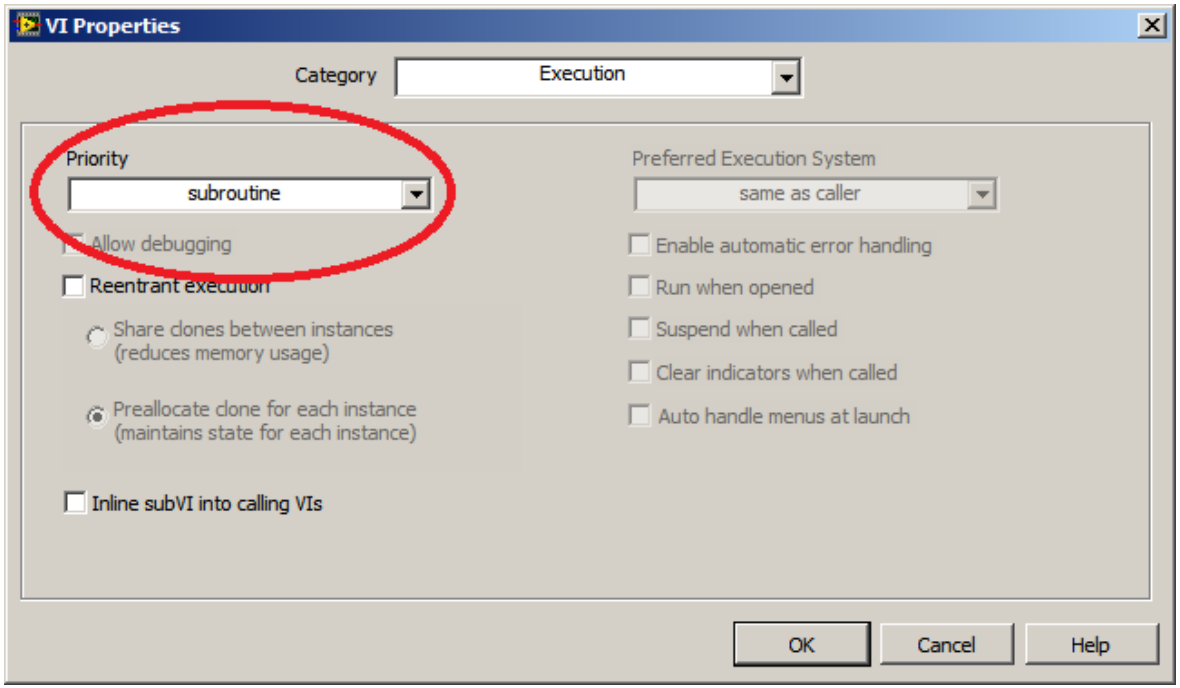
- Lo Mejor

??? – '*timing probe idea*'
DEMO!!

Técnicas de Programación

Tres sugerencias específicas sobre como mejorar el rendimiento de tus VIs sin tener que cambiar absolutamente nada del código

#1. Subroutine Priority



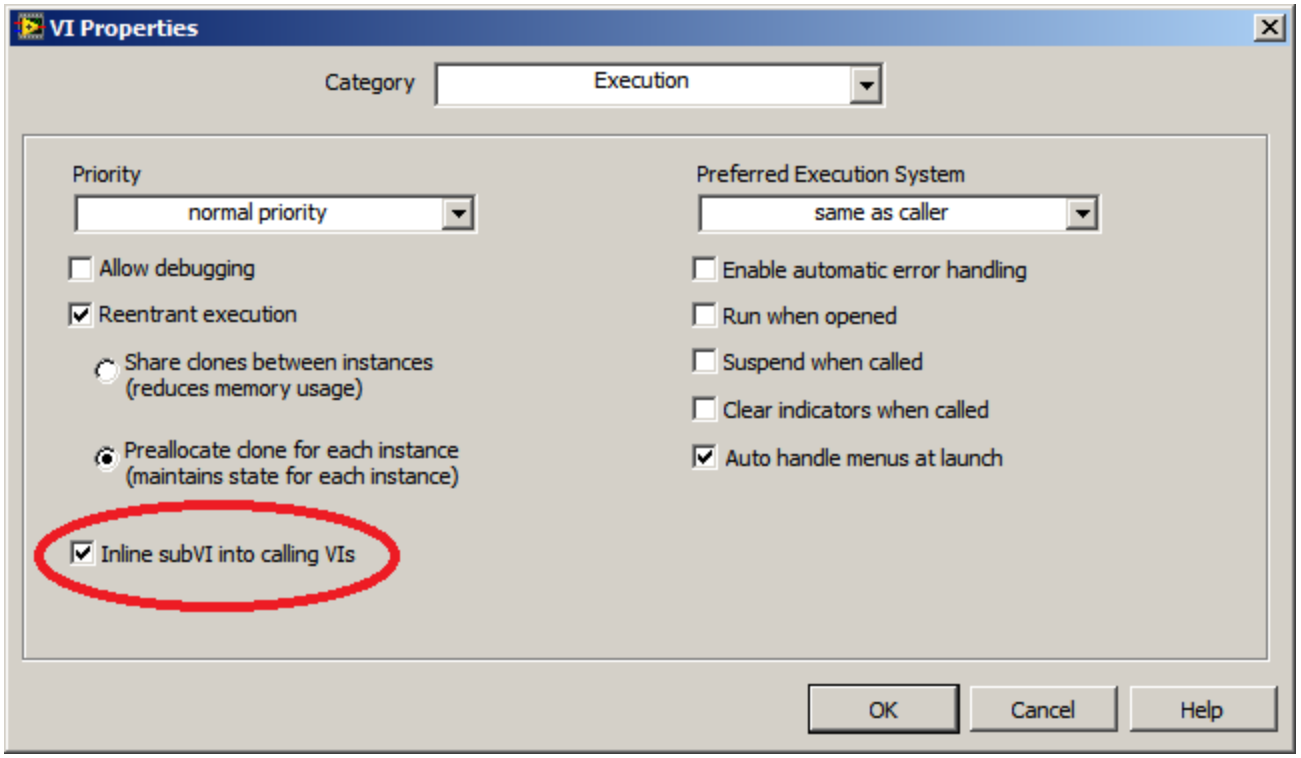
Al ponerle esta prioridad a un VI hace que éste tome control del hilo en el que se está ejecutando. Esto hace que se ejecute de la manera más eficiente posible.

#1. Subroutine Priority

A tener en cuenta

- Un VI con prioridad de subrutina solo puede llamar a otros VIs de este tipo
- No pueden llamar a nada que los bloquee (Wait, One Button Dialog, llamadas a VISA, etc.)
- Los controles e indicadores del Front Panel no se actualizan durante la ejecución de este tipo de VIs
- No puede haber ningún otro VI ejecutándose en el hilo del VI que lo llama
- ¡DEMO!

#2. Inlining SubVIs



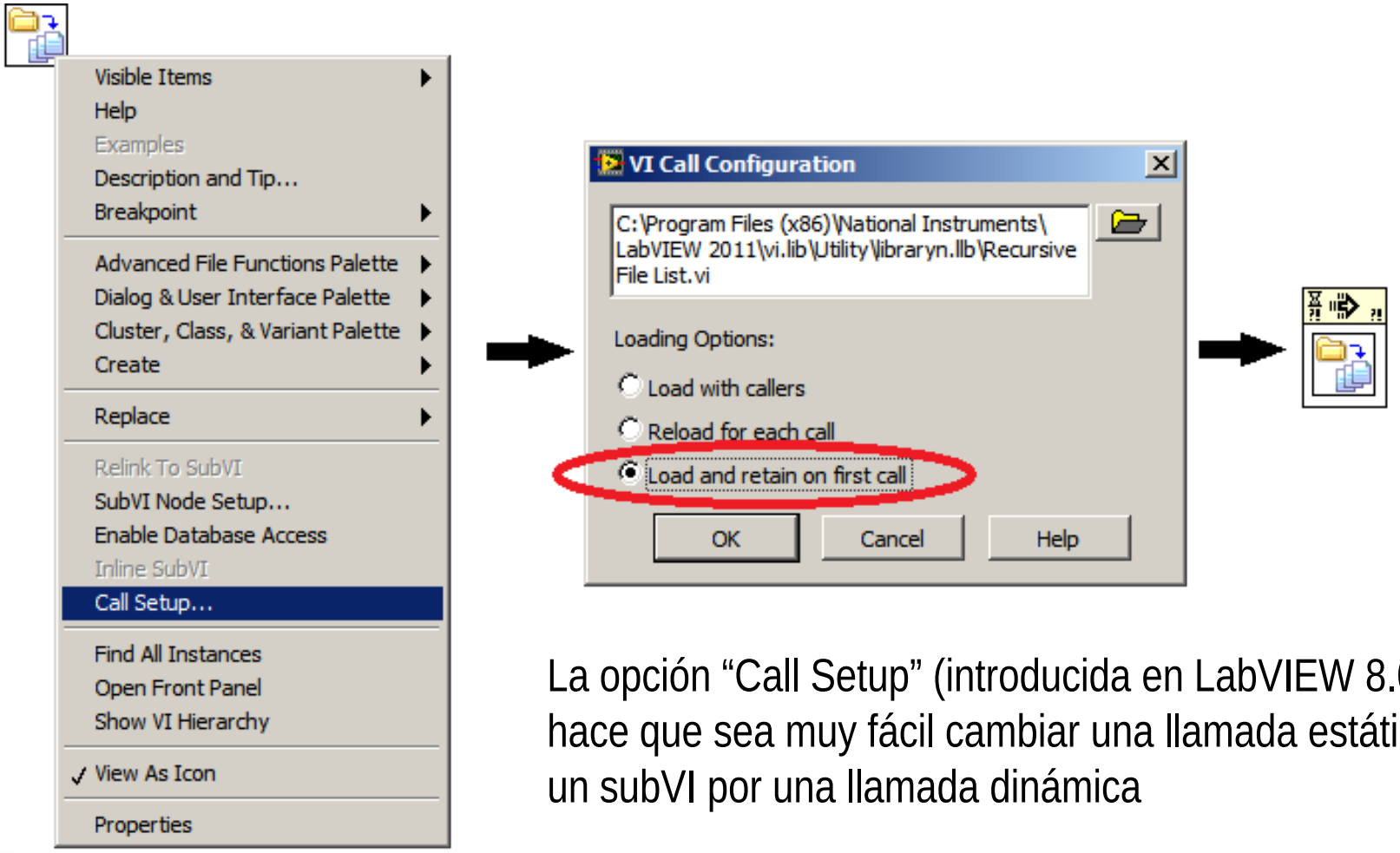
Desde LabVIEW 2010, **subVI inlining** elimina el retardo de llamada a un subVI, ejecutando el código de los subVIs como si fuera propio del VI que lo llama.

#2. Inlining SubVIs

A tener en cuenta

- El VI tiene que ser reentrante
- No se pueden debuguear
- Puede que el rendimiento baje si el VI en cuestión es grande
- No pueden contener llamadas recursivas
- No pueden tener ni Property ni Invoke Nodes
- DEMO!!!

#3. Easy Dynamic Calls



La opción “Call Setup” (introducida en LabVIEW 8.0) hace que sea muy fácil cambiar una llamada estática a un subVI por una llamada dinámica

#3. Easy Dynamic Calls

A tener en cuenta

- Si el VI llamante está en modo de edición, todos los Vis dinámicos estarán en memoria
- La ventana de configuración muestra un path absoluto, pero realmente se guarda uno relativo
- “Reload for each call” sólo debería usarse si se necesita liberar la memoria ocupada por cada llamada a un subVI
- DEMO!!