COME USARE LE VARIABILI CONDIVISE IN LABVIEW

Mike Trimborn

L'articolo introduce le variabili condivise e contiene una discussione sulle loro caratteristiche e prestazioni in LabVIEW

LabVIEW permette di accedere a un'ampia varietà di tecnologie per creare applicazioni distribuite. La variabile condivisa (shared variable) introdotta in LabVIEW 8 rappresenta un importante passo avanti nella semplificazione della programmazione necessaria per tali applicazioni.

CREAZIONE DELLE SHARED VARIABLE

Utilizzando le shared variable, potete condividere dati fra più cicli paralleli nello stesso diagramma o fra VI distinti distribuiti su una rete. A differenza di molti metodi esistenti di condivisione dei dati in LabVIEW come UDP/TCP, le code e le FIFO Real-Time, tipicamente dovete configurare la shared variable in fase di creazione della risorsa, mediante finestre di configurazione delle proprietà, e non è necessario includere codice di configurazione nella vostra applicazione.

Potete creare tre tipi di shared variable: single-process, network-published e time-triggered. L'articolo discute in dettaglio le shared variable single-process e network-published.

Per creare una shared variable dovete avere un progetto aperto. Per aggiungere una shared variable a un progetto, cliccate con il pulsante destro del mouse su un target, su una libreria di progetto o su una cartella all'interno di una libraria di progetto nella finestra **Project Explorer**. Quindi selezionate **New»Variable** nel menu rapido per visualizzare la finestra **Shared Variable Properties**, selezionate le opzioni di configurazione di vostro interesse e cliccate il pulsante **OK**.

Se cliccate con il pulsante destro del mouse direttamente su un target oppure su una cartella che non sia all'interno di una libreria di progetto e selezionate **New»Variable** dal menu rapido, LabVIEW crea una nuova libreria di progetto e pone la shared variable al suo interno.

La figura 1 mostra la finestra di dialogo **Shared Variable Properties** per una shared variable single-process. Il Modulo LabVIEW Real-Time e il Modulo LabVIEW Datalogging and Supervisory Control (DSC) offrono proprietà configurabili addizionali.

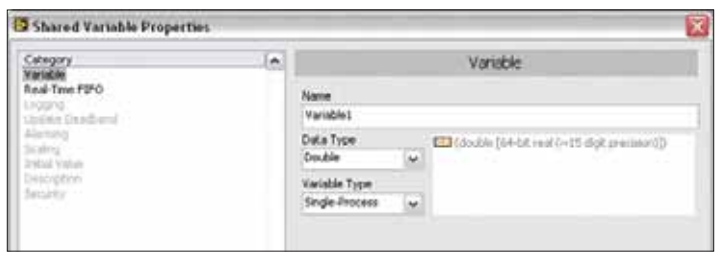

Figure 1 – Finestra delle proprietà di una shared variable single-process

TIPI DI DATO

Creata una nuova shared variable, potete selezionare fra un numero elevato di tipi di dato standard. In aggiunta, potete specificare anche un tipo di dato custom selezionando **Custom** nell'elenco a tendina **Data Type** e navigando verso un controllo custom. Tuttavia, quando si utilizzano tipi di dato custom, LabVIEW non consente di abilitare le opzioni FIFO real-time e scalatura. Inoltre, se avete installato il Modulo LabVIEW DSC, la gestione degli allarmi si limita alle notifiche di stato errato.

Dopo avere configurato le proprietà della shared variable ed avere cliccato il pulsante **OK**, la shared variable appare nella vostra finestra **Project Explorer** sotto la libreria o il target che avete selezionato, come mostrato in figura 2.

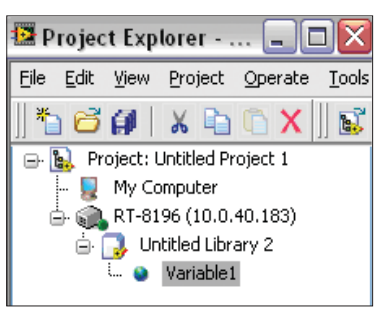

Figure 2 – La shared variable nel progetto

RIFERIMENTI ALLA VARIABILE

Dopo avere aggiunto una shared variable ad un progetto LabVIEW, potete trascinare la shared variable nello schema a blocchi di un VI per effettuare operazioni di lettura e scrittura, come mostrato in figura 3.

Potete cliccare con il pulsante destro del mouse su una shared variable nella finestra **Project Explorer** e così editarne le pro-

prietà in qualsiasi momento. Il progetto LabVIEW propaga le nuove impostazioni a tutti i riferimenti alla variabile presenti in memoria. Quando salvate la libreria di variabili, questi cambiamenti vengono applicati anche alla definizione della variabile memorizzata su disco.

SHARED VARIABLE SINGLE-PROCESS

Utilizzate variabili single-process per trasferire dati fra due punti del diagramma di uno stesso VI che non possono essere connessi con un filo, come accade nel caso di cicli paralleli, oppure fra due VI distinti presenti sulla stessa mecchina. L'implementazione di una variabile single-process è simile a quella di una variabile globale di LabVIEW. Il vantaggio principale della nuova variabile è la possibilità di convertirla in una variabile network-published, accessibile a qualsiasi nodo sulla rete.

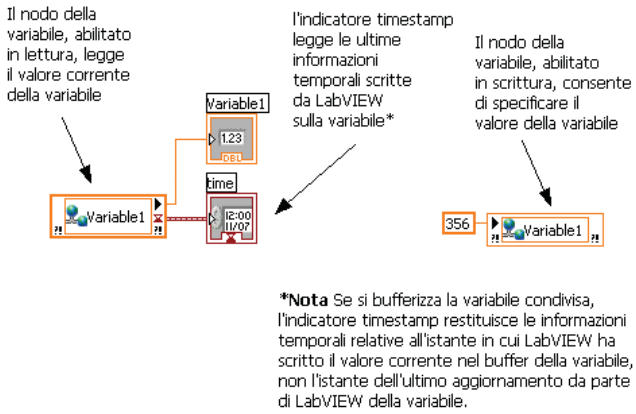

Figura 3 – Operazioni di lettura e scrittura di una shared variable

LE SHARED VARIABLE SINGLE-PROCESS E LABVIEW REAL-TIME

Per mantenere il determinismo, un'applicazione real-time richiede l'uso di un meccanismo deterministico non bloccante per trasferire i dati dalle parti critiche del codice, come i timed loop a priorità più alta e i VI con priorità time-critical, alle sezioni non deterministiche dell'applicazione. Quando installate il modulo LabVIEW Real-Time, potete configurare una shared variable per far sì che utilizzi le FIFO real-time, abilitando tale funzionalità dalla finestra di dialogo **Shared Variable Properties**. In questo modo potete evitare l'uso dei VI real-time FIFO di basso livello.

Figura 4 – Variabile condivisa operante come FIFO real-time

LabVIEW crea una FIFO real-time la prima volta che un nodo di variabile cerca di scrivere o leggere una variabile. Questo comportamento si traduce in un tempo di esecuzione leggermente più lungo al primo utilizzo di una shared variable rispetto agli accessi effettuati successivamente. Se un'applicazione richiede una temporizzazione estremamente precisa, vi conviene prevedere alcune iterazioni iniziali di assestamento, per tenere conto di questa fluttuazione nei tempi di accesso, oppure leggere la variabile almeno una volta all'esterno del ciclo critico (fig. 4).

LabVIEW crea un'unica FIFO real-time per ogni shared variable single-process anche quando la variabile gestisce lettori/scrittori multipli. Per assicurare l'integrità dei dati, gli scrittori si bloccano reciprocamente e lo stesso accade per i lettori multipli. Tuttavia, un lettore non blocca uno scrittore ed uno scrittore non blocca un lettore. National Instruments raccomanda di evitare lettori/scrittori multipli di shared variable single-process utilizzate in cicli time-critical (fig. 5).

Potete selezionare fre due tipi leggermente differenti di variabile con FIFO real-time abilitata: il buffer singolo elemento e il buffer multielemento. Una prima differenza fra i due è che il buffer singolo elemento non riporta notifiche relative a condizioni di overflow o underflow. Una seconda distinzione si riferisce al valore restituito da LabVIEW quando più lettori leggono un buffer vuoto. I lettori multipli di una FIFO singolo elemento ricevono lo stesso valore e il buffer restituisce lo stesso valore finché uno scrittore scrive nuovamente su quella variabile. Ciascuno dei lettori multipli di una FIFO multielemento vuota ottiene l'ultimo valore che ha letto dal buffer, oppure il valore di default

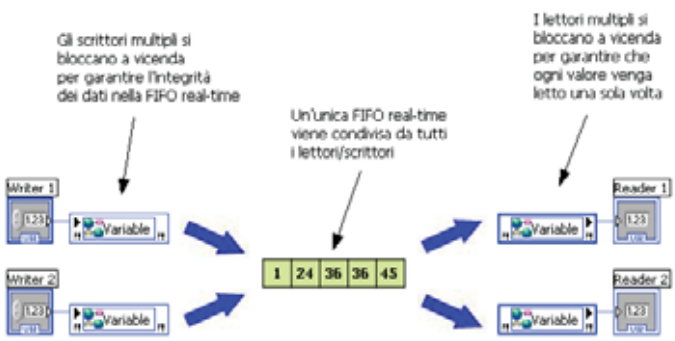

Figura 5 – Scrittori e lettori multipli che condividono una singola FIFO

Figura 6 – Shared variable con FIFO real-time multielemento e modalità di lettura del buffer condiviso da parte di più variabili

associato al tipo di dato della variabile, nel caso in cui non abbia mai letto prima la variabile. Tale comportamento è descritto in figura 6.

Se un'applicazione richiede che ogni lettore ottenga ogni singolo dato scritto in una shared variable con FIFO multielemento, utilizzate una shared variable separata per ciascun lettore.

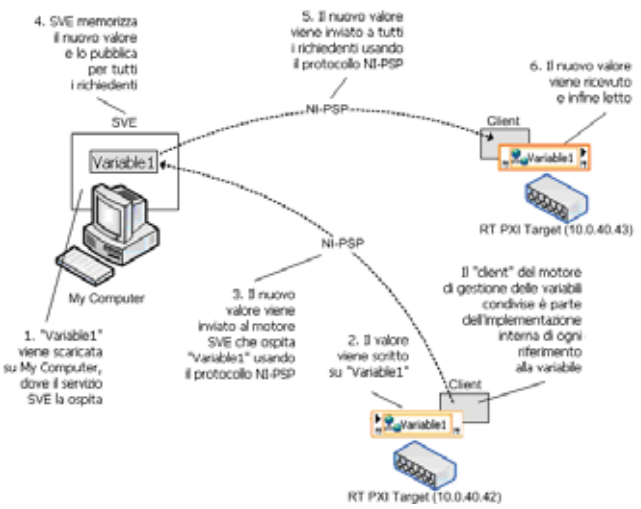

Figura 7 – Motore delle shared variable e gestione dei cambi di valore delle variabili

SHARED VARIABLE NETWORK-PUBLISHED

Usando una shared variable network-published, cioè pubblicata in rete, potete scrivere e leggere una variabile per condividere dati su una rete Ethernet. L'implementazione delle modalità di comunicazione sulla rete viene gestita interamente dalla variabile.

Oltre a rendere disponibili i vostri dati sulla rete, la shared variable network-published aggiunge molte funzionalità non disponibili con la shared variable single-process; questo comporta che l'implementazione interna della variabile pubblicata in rete sia molto più complessa. I prossimi paragrafi ne discutono alcuni aspetti e offrono raccomandazioni per ottenere le migliori prestazioni con il loro utilizzo.

IL PROTOCOLLO NI-PSP

Le variabili condivise network-published utilizzano NI Publish-Subscribe Protocol (NI-PSP) per inviare e ricevere dati attraverso la rete. Il protocollo NI-PSP è costruito sopra il protocollo UDP e ne sfrutta quindi i tipici aspetti di rapidità ed efficienza. l protocollo NI-PSP utilizza meno ampiezza di banda ed è più efficiente del TCP/IP, in relazione alle specifiche proprie del protocollo NI-PSP. Tuttavia, a differenza del protocollo UDP, NI-PSP garantisce la ricezione dei dati, implementando uno strato addizionale sopra il protocollo UDP grezzo.

DISTRIBUZIONE E HOSTING DELLE VARIABILI

Dovete **assegnare** le shared variable pubblicate in rete a un motore di gestione delle variabili condivise, o SVE (Shared Variable Engine), che **ospita** (hosting) i valori delle variabili sulla rete.

Quando scrivete su un nodo di variabile, LabVIEW invia il nuovo valore al motore di competenza che ospita la variabile. Il Variable Engine **pubblica** quindi il valore in modo tale che i richiedenti ottengano il valore aggiornato.

La figura 7 illustra tale processo. Utilizzando la terminologia client/server, il motore SVE è il ***server*** per una shared variable, mentre tutti i riferimenti alla variabile sono i ***client***, indipendentemente dal fatto che leggano o scrivano la variabile. Il lato client è parte integrante dell'implementazione di ogni nodo di variabile condivisa.

LE VARIABILI NETWORK-PUBLISHED E LABVIEW REAL-TIME

Potete abilitare la FIFO real-time anche per una shared variable pubblicata in rete ma occorre tenere presente che si comportano in modo diverso rispetto alle variabili single-process, come visto in precedenza. Con le variabili single-process, tutti gli scrittori e tutti i lettori condividono un'unica FIFO real-time. Con le variabili network-published, ogni lettore possiede una propria FIFO real-time, sia nel caso a singlo elemento che nel caso multielemento, come illustrato in figura 8.

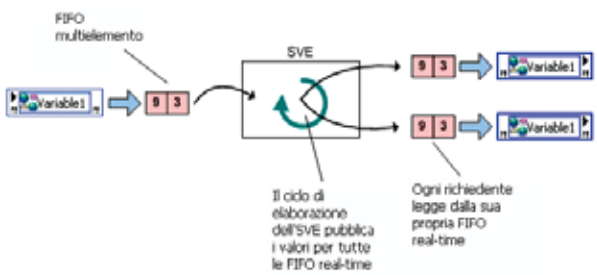

Figura 8 – Shared variable pubblicata in rete con opzione FIFO real-time abilitata

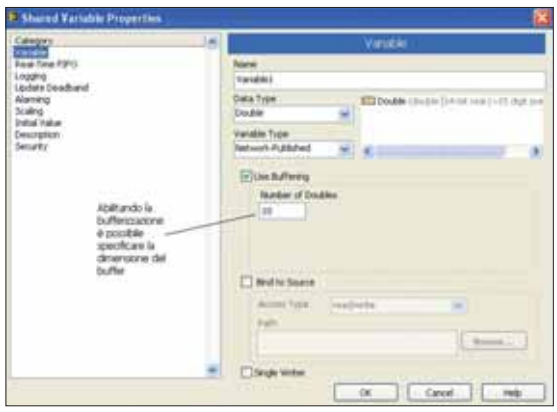

Figura 9 – Abilitazione della bufferizzazione di una shared variable pubblicata in rete

BUFFERIZZAZIONE DI RETE

Con le shared variable network-published potete abilitare un buffer di rete, configurandolo nella finestra di dialogo **Shared Variable Properties**, come illustrato in figura 9. Con la bufferizzazione, tenete sotto controllo le fluttuazioni temporanee che possono avvenire fra le velocità di lettura e scrittura di una variabile. I lettori che occasionalmente leggono una variabile più lentamente rispetto alla velocità di scrittura, possono perdere qualche aggiornamento. Se l'applicazione può tollerare la perdita occasionale di qualche valore, non c'è necessità di abilitare la bufferizzazione. Tuttavia, se il lettore deve ricevere ogni aggiornamento, abilitate la bufferizzazione. Potete impostare le dimensioni del buffer nella pagina **Variable** della finestra di dialogo **Shared Variable Properties**, in modo tale da poter decidere quanti aggiornamenti l'applicazione deve mantenere memorizzati in coda prima di iniziare a sovrascrivere i vecchi dati. LabVIEW usa la dimensione del buffer che specificate per creare due buffer interni, uno sul motore SVE e uno a livello del richiedente.

Dato che ogni lettore di una shared variable pubblicata in rete ottiene il proprio buffer, i lettori non interagiscono fra loro.

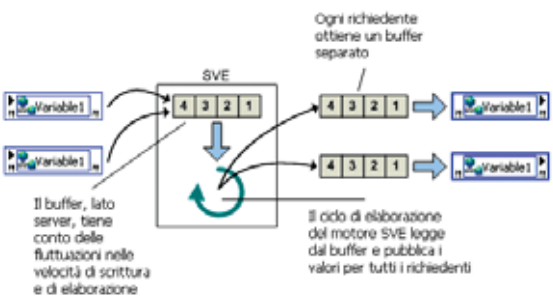

Figura 10 – Bufferizzazione di rete

La bufferizzazione è utile solo nelle situazioni in cui le velocità di lettura/scrittura hanno fluttuazioni temporanee. Se l'applicazione viene eseguita per un periodo indefinito, i lettori che leggono costantemente ad una velocità più bassa del rate di scrittura alla fine perdono dei dati, indipendentemente dalle dimensioni del buffer che specificate. Poiché la bufferizzazione alloca un buffer per ogni richiedente, usatela solo quando è necessario, per evitare di sprecare memoria.

BUFFERIZZAZIONE DI RETE E FIFO REAL-TIME

Se abilitate sia la bufferizzazione di rete che la FIFO real-time, l'implementazione della shared variable includerà sia un buffer di rete che una FIFO real-time. Ricordate che l'opzione FIFO real-time crea una coda separata per ogni scrittore e ogni lettore; pertanto scrittori e lettori multipli non si bloccano reciprocamente. Anche se potete impostare indipendentemente le dimensioni di questi due buffer, nella maggior parte dei casi National Instruments raccomanda di utilizzare le stesse dimensioni per entrambi (fig. 11).

VITA DEI BUFFER

LabVIEW crea i buffer di rete e le FIFO real-time in occasione di una scrittura o lettura iniziale. I buffer lato server sono creati quando uno scrittore scrive per la prima volta su una shared variable. I buffer lato client sono creati quando un richiedente legge per la prima volta da una shared variable. Se uno scrittore scrive dati su una shared variable prima che un richiedente inizi a leggere da tale variabile, i valori iniziali non saranno più disponibili per il richiedente (fig. 12).

OVERFLOW/UNDERFLOW DEL BUFFER

La shared variable pubblicata in rete riporta le condizioni di overflow e underflow dei buffer di rete a partire dalla versione di LabVIEW 8.20. Una FIFO real-time, invece, indica l'overflow/underflow della coda, restituendo errori, in tutte le versioni di LabVIEW. Un'applicazione in LabVIEW 8.0 o 8.0.1 può verificare gli underflow dei buf-

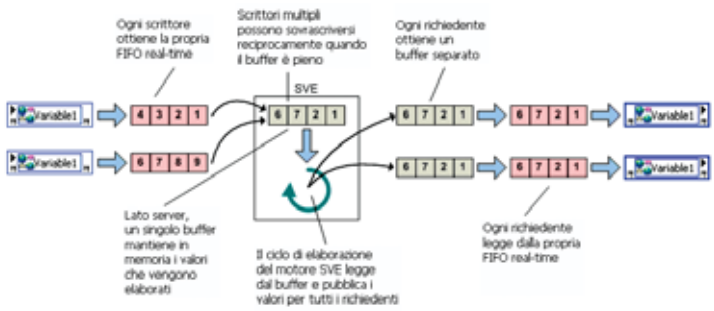

Figura 11 – Bufferizzazione di rete e FIFO real-time abilitata

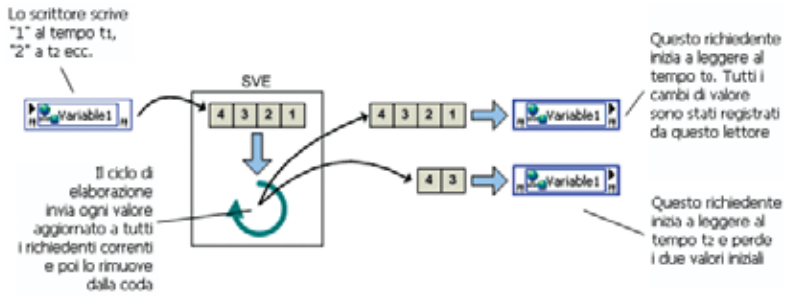

Figura 12 – Vita dei buffer

fer di rete in due modi. Quando aggiornate la variabile a una frequenza inferiore a 1 kHz, tenuto conto che la risoluzione temporale del timestamp di una shared variable è di 1 ms, potete confrontare il timestamp di una variabile con quello relativo a una lettura successiva per rilevare gli underflow dei buffer. Oppure, per applicazioni non real-time, il lettore può utilizzare un numero sequenziale da legare ai dati per notificare gli overflow/underflow dei buffer. Non potete utilizzare il secondo approccio con le shared variable usate all'interno di un loop time-critical, perché le shared variable con opzione FIFO real-time abilitata non supportano il tipo di dato Custom Control (cluster).

VITA DELLE SHARED VARIABLE

Come detto in precedenza, tutte le shared variable sono parte di una libreria di progetto. Il motore SVE registra le librerie di progetto e le shared variable contenute in queste librerie ogni volta che LabVIEW richiede una di tali variabili. Di default, l'SVE scarica e pubblica una libreria di shared variable sul target di competenza, come indicato dal progetto, non appena eseguite un VI che fa riferimento ad una qualsiasi delle variabili contenute. Il motore scarica l'intera libreria e pubblica tutte le shared variable presenti, indipendentemente dal fatto che un VI in esecuzione faccia riferimento solamente a una parte di esse. Potete scaricare manualmente qualsiasi libreria di progetto cliccando con il pulsante destro del mouse sulla libreria nella finestra **Project Explorer**.

L'arresto del VI o il reboot della macchina che ospita la shared variable non rende la variabile indisponibile alla rete. Se volete rimuovere la shared variable dalla rete, dovete farlo esplicitamente nella finestra **Project Explorer**. Potete anche selezionare **Tools»Shared Variable»Variable Manager** per rimuovere variabili o intere librerie di variabili.

DATA BINDING SUL PANNELO FRONTALE

Una caratteristica addizionale disponibile solo per le shared variable pubblicate in rete è il data binding per oggetti del pannelo frontale. Dalla finestra **Project Explorer** trascinate una shared variable sul pannello frontale di un VI per creare un controllo legato alla shared variable. Quando abilitate il data binding per un controllo, cambiando il valore del controllo cambiate il valore della shared variable alla quale il controllo è legato. Mentre il VI è in esecuzione, se la connessione all'SVE ha successo, appare un piccolo indicatore verde accanto all'oggetto sul pannello frontale del VI, come si vede in figura 13.

Potete modificare le proprietà di binding relative a un qualsiasi controllo o indicatore dalla pagina **Data Binding** della finestra di dialogo **Properties**. Quando utilizzate il modulo LabVIEW Real-Time o LabVIEW DSC, potete selezionare **Tools»Shared Variable»Front Panel Binding Mass Configuration** per visualizzare la finestra di dialogo **Front Panel Binding Mass Configuration** e creare un'interfaccia operatore che lega numerosi controlli e indicatori alle shared variable.

ACCESSO PROGRAMMATICO

Come discusso in precedenza, potete creare, configurare e scaricare interattivamente le shared variable utilizzando il LabVIEW Project, e potete leggere e scrivere le variabili utilizzando i nodi relativi presenti sullo schema a blocchi o attraverso il data binding. LabVIEW offre anche l'accesso programmatico a tutte queste funzionalità. Utilizzate il VI Server per creare programmaticamente librerie di progetto e shared variable nelle applicazioni dove dovete creare un grande numero di variabili. Inoltre, il Modulo LabVIEW DSC offre un set completo di VI per creare ed editare programmaticamente shared variable e librerie di progetto, nonché per gestire l'SVE. Potete creare programmaticamente librerie di shared variable solo su sistemi Windows; tuttavia, potete distribuire programmaticamente queste nuove librerie sia su sistemi Windows che LabVIEW Real-Time. Usate l'API DataSocket nelle applicazioni dove dovete cambiare dinamicamente la shared variable che un VI legge e scrive. Potete cambiare dinamicamente una shared variable costruendo programmaticamente l'URL della connessione DataSocket. Potete anche controllare programmaticamente altre caratteristiche della shared variable, come le dimensioni del buffer. Inoltre, l'API DataSocket ha una lettura

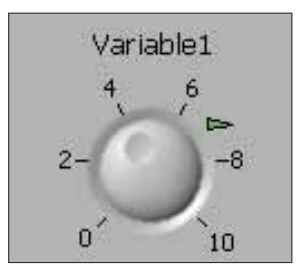

Figura 13 – Binding di un controllo a una shared variable

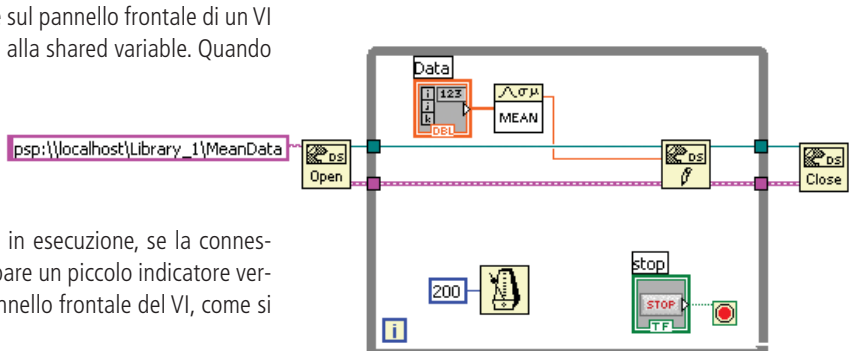

Figura 14 – Uso dell'API DataSocket per leggere e scrivere programmaticamente shared variable

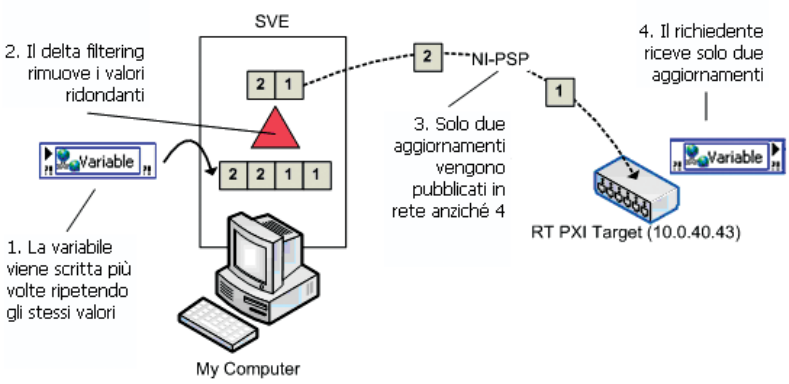

Figura 15 – Architettura idonea per un delta filtering efficace

bloccante che restituisce un valore solo quando la shared variable viene aggiornata (fig. 14).

SHARED VARIABLE ENGINE

L'SVE è un framework che permette ad una shared variable pubblicata in rete di inviare valori attraverso la rete. Su Windows, LabVIEW configura l'SVE come servizio e lancia l'SVE allo startup di sistema. Su un target real-time, l'SVE è un componente di startup installabile che viene caricato al boot di sistema. Per utilizzare le shared variable pubblicate in rete, un SVE deve essere in esecuzione su almeno uno dei nodi del sistema distribuito. Qualsiasi nodo sulla rete può leggere o scrivere le variabili pubblicate dall'SVE. Potete anche avere più SVE installati su più sistemi simultaneamente se dovete distribuire le shared variable in posizioni differenti in base alle esigenze applicative.

RACCOMANDAZIONI PER LA SCELTA DEL LUOGO IN CUI OSPITARE LE VARIABILI

Quando scegliete il dispositivo che ospiterà le variabili di rete e da cui effettuare la distribuzione delle variabili che utilizzate in un sistema distribuito, dovete considerare una serie di fattori.

Il dispositivo di elaborazione è compatibile con l'SVE?

La tabella 1 riassume le piattaforme per le quali è disponibile l'SVE e identifica le piattaforme che possono utilizzare variabili condivise pubblicate in rete tramite nodi di variabile o l'API DataSocket. National Instruments richiede 32 MB di RAM e ne raccomanda 64 MB per l'SVE su tutte le piattaforme valide.

L'applicazione richiede funzionalità di datalogging e supervisione?

Se volete usare le caratteristiche proprie del modulo LabVIEW DSC, dovete ospitare le shared variable su Windows. Il modulo LabVIEW DSC aggiunge le seguenti funzionalità alle shared variable pubblicate in rete:

- Logging storico nel database NI Citadel.
- Allarmi in rete e logging degli allarmi.
- Scalatura.
- Sicurezza basata sull'utente.
- Valore iniziale.
- Possibilità di creare server di I/O custom.
- Integrazione della struttura a eventi di LabVIEW con le shared variable.
- VI LabVIEW per controllare programmaticamente tutti gli aspetti delle shared variable e del motore delle variabili. Questi VI sono particolarmente utili per gestire grandi numeri di shared variable.

Il dispositivo di elaborazione ha risorse adeguate in termini di processore e memoria?

L'SVE è un processo addizionale che richiede risorse di elaborazione e memoria. Per ottenere le migliori prestazioni in un sistema distribuito, installate l'SVE su macchine che garantiscono la massima capacità di memoria ed elaborazione in quel sistema.

Dove vengono generati ed elaborati i dati?

L'SVE riduce il traffico di rete quando la bufferizzazione è disabilitata. Abilitate la bufferizzazione se vi occorre ogni nuovo valore di una shared variable. Quando la bufferizzazione è disabilitata, l'SVE esegue il cosiddetto **delta filtering**: rimuove i valori ridondanti e invia ai richiedenti solo gli effettivi cambi di valore di una shared variable.

L'architettura di un sistema distribuito può influenzare notevolmente l'efficacia di questa funzionalità, come si vede in figura 15. LabVIEW implementa questo tipo di filtraggio solo per tipi scalari come numeri interi, numeri decimali in virgola mobile e booleani, e per gli array di stringhe e di booleani.

Il delta filtering è efficace soprattutto se generate dati sulla stessa macchina che ospita una variabile, come mostra-

	PC Windows	Mac OS	Linux	PXI Real-Time	Compact FieldPoint	CompactRIO	Compact Vision System	PC commerciali con LabVIEW Real-Time ETS	RTX
SVE	✓	•	•	✓	✓	✓	✓	✓	✓
Nodi di variabile	✓	•	•	✓	✓	✓	✓	✓	✓
DataSocket API con PSP	✓	•	•	✓	✓	✓	✓	✓	✓

Tabella 1 – Sintesi delle compatibilità per le shared variable pubblicate in rete

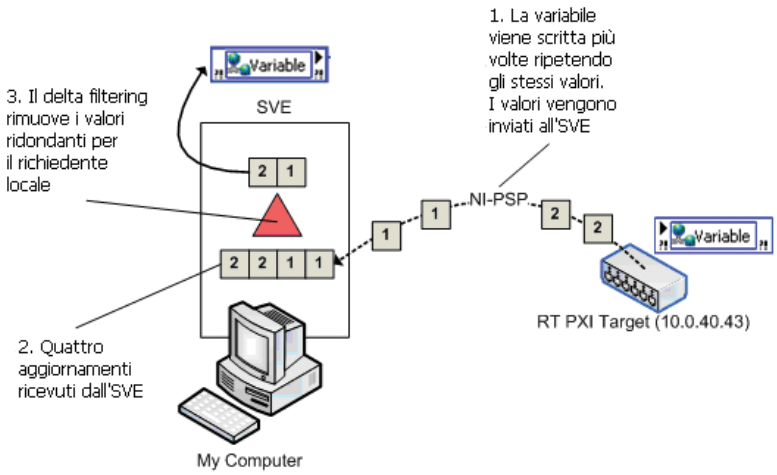

Figura 16 – Architettura che non consente un delta filtering efficace

to in figura 15. In questo esempio, lo scrittore è su **My Computer** e scrive più valori ridondanti su una variabile. L'SVE esegue il filtraggio delta e invia solo due aggiornamenti sulla rete al target PXI RT.

La figura 16 illustra un esempio di filtraggio delta non efficace. Lo scrittore è sul target PXI RT e invia all'SVE tutti i cambi di valore, inclusi gli aggiornamenti ridondanti.

I valori ridondanti vengono rimossi successivamente su **My Computer** per i richiedenti ma questo non riduce il traffico in rete. Se costruite un'applicazione distribuita e sapete che alcuni dei sistemi possono andare offline periodicamente, fate in modo di ospitate l'SVE su un sistema che sia sempre online.

CARATTERISTICHE ADDIZIONALI DELLO SHARED VARIABLE ENGINE

La figura 17 illustra le numerose responsabilità dell'SVE, oltre a quella di gestire le shared variable pubblicate in rete:

- Raccogliere i dati ricevuti dai server di I/O.
- Fornire i dati ai richiedenti, attraverso i server OPC e PSP.
- Fornire servizi di scalatura, allarme e logging per qualsiasi shared variable per cui siano stati configurati tali servizi che, ricordiamo, sono disponibili solo con il modulo LabVIEW DSC.
- Monitorare le condizioni di allarme e rispondere di conseguenza.

SERVER DI I/O

I server di I/O sono plug-in dell'SVE con i quali i programmi possono usare l'SVE per pubblicare dati. NI FieldPoint 5.0 include un server di I/O che pubblica i dati direttamente da banchi FieldPoint sull'SVE. Poiché l'SVE è un server OPC, la combinazione dell'SVE e del server di I/O FieldPoint configura di fatto un server OPC per FieldPoint. L'installer del driver FieldPoint non include l'SVE, che deve essere installato da un altro componente sofware, ad esempio LabVIEW.

Anche il driver NI-DAQmx dalla versione 8.0 include un server di I/O che può pubblicare automaticamente canali virtuali globali NI-DAQmx sull'SVE. Tale server di I/O sostituisce il server OPC e la tecnologia di accesso remoto RDA (Remode Device Access) presenti nel vecchio driver NI-DAQ Traditional.

Il driver NI-DAQmx include l'SVE. Con il modulo LabVIEW DSC, gli utenti possono creare nuovi server di I/O.

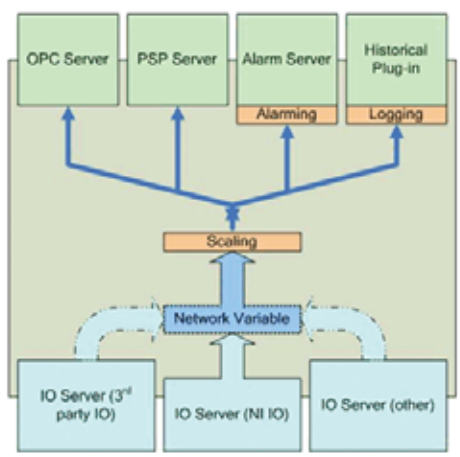

Figura 17 – Shared Variable Engine (SVE)

OPC

L'SVE è conforme alla versione 3.0 e può operare come un server OPC su macchine Windows. Qualsiasi client OPC può scrivere o leggere una shared variable ospitata su una macchina Windows. Quando installate il modulo LabVIEW DSC su una macchina Windows, l'SVE può operare anche come client OPC. Potete legare le shared variable ospitate su una macchina Windows a degli item OPC con il DSC e abilitare la comunicazione in lettura e scrittura tra le variabili e gli item OPC.

Poiché OPC è una tecnologia basata su COM, una API di Windows, i target real-time non lavorano direttamente con OPC. Come mostrato in figura 18, potete comunque accedere agli item OPC da un target real-time ospitando le shared variable su

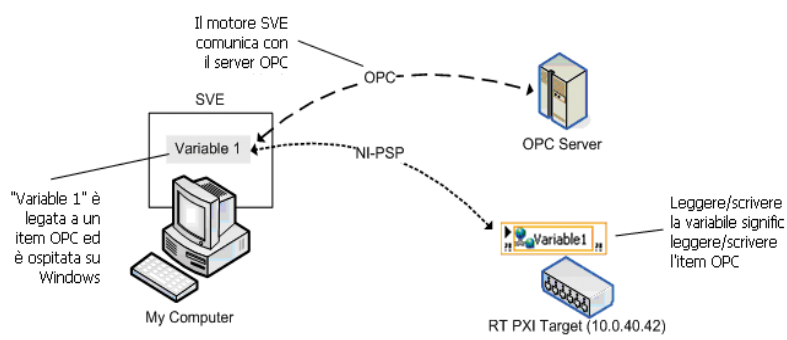

Figura 18 – Legare variabili ad item OPC

una macchina Windows.

PRESTAZIONI

Questo paragrafo fornisce delle linee guida generali per creare applicazioni ad alte prestazioni utilizzando le shared variable. Poiché la shared variable single-process ha un'implementazione simile alle variabili globali e alle FIFO real-time di LabVIEW, National Instruments non ha particolari raccomandazioni per ottenere buone prestazioni con questo tipo di variabili. I paragrafi seguenti si concentrano quindi sulle shared variable pubblicate in rete.

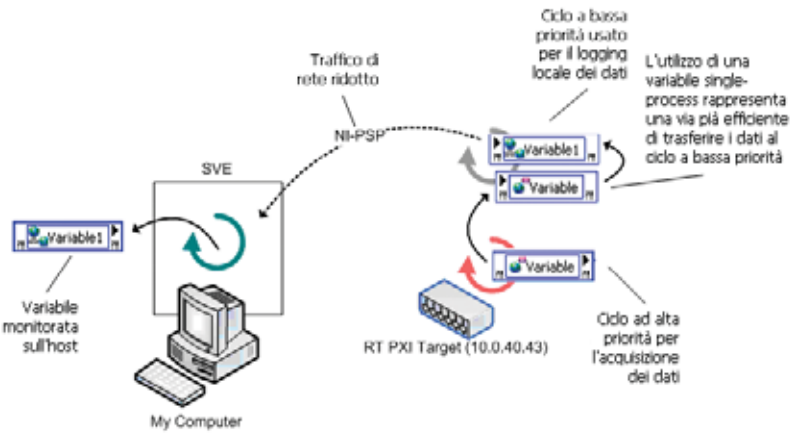

Figura 20 – Uso efficiente delle shared variable pubblicate in rete su un target real-time

NON MONOPOLIZZATE IL PROCESSORE

La shared variable pubblicata in rete semplifica i diagrammi a blocchi di LabVIEW nascondendo all'interno della sua implementazione molti dettagli strutturali tipici della programmazione di rete. Le applicazioni comprendono i VI LabVIEW, l'SVE e il codice client SVE. Per ottenere le migliori prestazioni dalle shared variable, sviluppate l'applicazione in modo tale che liberi regolarmente il processore per l'esecuzione dei thread SVE. Un modo per ottenere questo effetto è quello di inserire delle attese nei cicli di elaborazione e assicurarsi che l'applicazione non faccia uso di cicli non temporizzati. L'esatta quantità di tempo che dovete attendere dipende dall'applicazione, dal processore e dalla rete; ogni applicazione richiede un certo livello di messa a punto empirica per ottenere le massime prestazioni.

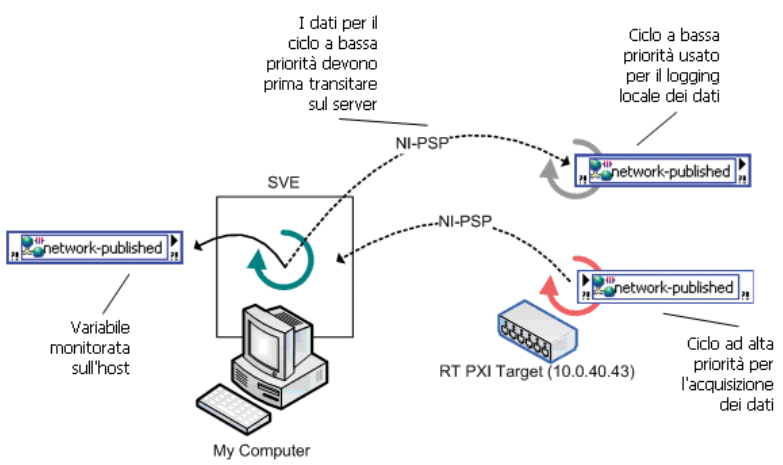

Figura 19 – Uso inefficiente delle shared variable pubblicate in rete su un target real-time

CONSIDERATE LA POSIZIONE DELL'SVE

Nel paragrafo *Raccomandazioni per la scelta del luogo in cui ospitare le variabili* abbiamo discusso una serie di fattori che dovete considerare nella scelta del punto in cui installare l'SVE. La figura 19 mostra un altro fattore che può avere un forte impatto sulle prestazioni delle shared variable. Questo esempio riguarda un target real-time, ma le stesse considerazioni valgono anche per sistemi non real-time. La figura mostra un uso inefficiente delle shared variable pubblicate in rete: generate dati su un target real-time e avete necessità di eseguire il logging locale dei dati elaborati e monitorare il processo da una macchina remota.

Dato che i richiedenti delle variabili devono ricevere i dati dall'SVE, la latenza fra la scrittura nel loop ad alta priorità e la lettura nel loop a priorità normale è elevata e sono richiesti due viaggi attraverso la rete.

La figura 20 illustra un'architettura migliore per questa applicazione. L'applicazione usa una shared variable single-process per trasferire i dati fra il loop ad alta priorità e il loop a bassa priorità, riducendo notevolmente la latenza. Il loop a bassa priorità esegue il logging dei dati e scrive l'aggiornamento in una shared variable pubblicata in rete per il richiedente presente sull'host.

Note sull'autore

Mike Trimborn, LabVIEW FPGA Product Marketing Manager, National Instruments.