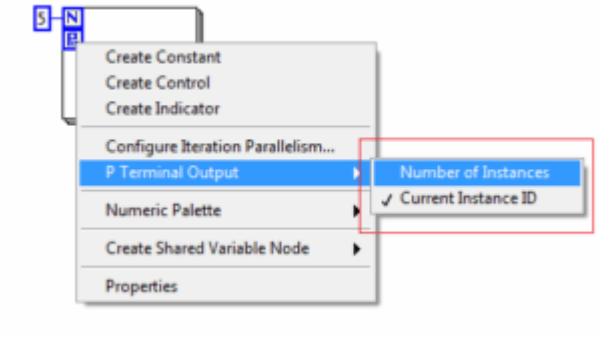
Parallel For loop LabVIEW

Category: LabVIEW Talk
scritto da admin | Gennaio 9, 2010



Parallel For Loop in LabVIEW: Quando è utile… e quando no?

A partire da LabVIEW 2009, è disponibile la possibilità di eseguire in parallelo le iterazioni di un ciclo For, sfruttando la potenza dei processori multicore. Una funzionalità potente, ma che richiede consapevolezza per essere utilizzata in modo efficace.

Anche nelle versioni più recenti di LabVIEW, come 2023 Q3, la gestione del parallelismo nei For Loop non è abilitata automaticamente, e per una buona ragione: parallelizzare non è sempre la scelta giusta.

Come si attiva

Per abilitare il parallelismo su un For Loop:

1. Fai clic con il tasto destro sul bordo del ciclo.

2. Seleziona Configure Iteration Parallelism.

3. Imposta il numero massimo di istanze parallele (se non specificato, LabVIEW cercherà di ottimizzarlo in base ai core disponibili).

A questo punto, LabVIEW suddivide le iterazioni in “chunk” da eseguire su thread distinti, se possibile.

Perché non sempre conviene parallelizzare?

Parallelizzare un For Loop non significa automaticamente migliorare le prestazioni. Anzi, in alcuni casi può addirittura peggiorarle.

Ecco alcuni aspetti da considerare:

Dipendenze tra iterazioni
Se le iterazioni non sono indipendenti (ad esempio accedono a una stessa risorsa o variabile globale), il parallelismo può causare race condition o

comportamenti non deterministici.

Overhead di scheduling
Per cicli con poche iterazioni o elaborazioni molto leggere, il tempo speso da LabVIEW per organizzare i thread può superare il tempo guadagnato con l'esecuzione parallela.

Accesso concorrente a dati o I/O
Se ogni iterazione accede a una risorsa condivisa (file, device, bus), il parallelismo può introdurre colli di bottiglia o errori di sincronizzazione.

Cosa fa internamente LabVIEW?

Quando abiliti il parallelismo, LabVIEW genera automaticamente più istanze del ciclo, ciascuna con il proprio contesto di esecuzione. Il compilatore (LLVM backend) si occupa di gestire l'assegnazione dei chunk alle CPU core disponibili.

Ma il compilatore non analizza automaticamente la sicurezza o l'efficienza del parallelismo: questa responsabilità resta al progettista del codice.

Quando ha senso usarlo?

Il ciclo esegue molte iterazioni indipendenti.

Le operazioni di ogni iterazione sono computazionalmente pesanti.

Non ci sono risorse condivise tra le iterazioni.

Si vogliono sfruttare al massimo le architetture multicore disponibili.In questi casi, abilitare il parallelismo può portare a un guadagno reale di prestazioni, anche significativo.

Approfondimenti

Spiegazioni dettagliate potete trovarle qui
–> http://zone.ni.com/devzone/cda/tut/p/id/9393

Un articolo su expression flow, al proposito http://expressionflow.com/2009/09/15/limits-of-for-loop-parallelism-how-parallel-they-really-execute/.