Practicas de Multiprocesadores

Primavera 2007

Paralelización con OpenMP (2)

Objetivo de la practica

Paralelizar con OpenMP el programa queens que calcula el número total de soluciones al problema de colocar n reinas en un tablero de ajedrez de nxn celdas sin que se maten entre ellas. El objetivo de reducir su tiempo de ejecución y obtener una buena escalabilidad al aumentar el número de procesadores. Para esta aplicación se utilizará una extensión ofrecida en el compilador de intel icc al modelo de programación OpenMP que permite la especificación de tareas[1].

Al final de la práctica, debereis entregar a través del "Racó de la FIB" las versiones paralelas del código junto con gráficas de escalabilidad y las observaciones que considereis necesarias.

Material para la practica

Copiar `../acmp/paral_OpenMP.tar`. Los ficheros `Makefile` y `run.sh` son muy similares a los utilizados en las otras prácticas de OpenMP (se compila con el flag `-openmp` y al ejecutar hay que especificar en la variable de entorno `OMP_NUM_THREADS` el número de threads a utilizar). Para la ejecución del programa, debereis utilizar `./queens -n<size>`, siendo `<size>` el número de reinas (y por lo tanto el tamaño del problema).

Paralelización de queens con paralelismo anidado

En primera opción, paralelizaremos el programa queens utilizando las construcciones convencionales que ya conoceis (parallel, for, …).

PREGUNTAS:

P1. Modificar el programa principal para explotar el paralelismo que existe en la aplicación (en el Makefile supone que el nombre del fichero con el código fuente es `queens-nested.c`). Pensad si es necesario limitar el número de niveles de paralelismo que se abren, y en caso afirmativo, incluid el mecanismo de control para permitirlo.

P2. Dibujar la gráfica de escalabilidad para la ejecución paralela utilizando entre 2 y 16 procesadores.

Extensión al modelo de programación

El compilador de Intel extendiende el modelo de programación OpenMP con dos pragmas que permiten la especificación de tareas a ejecutar por los procesadores que constituyen el "team" de threads asociados a la región paralela:

- `#pragma intel omp taskq`: identifica la sección de código dentro de la cual se van a generar tareas, utilizando el pragma que se explica en el siguiente punto. Dicha sección de código la va a ajecutar sólo uno de los threads vinculados a la región. El resto de threads quedan a la espera de que se generen tareas y se encolen en la cola asociada al `taskq`.

---

[1] Flexible Control Structures for Parallelism in OpenMP. Sanjiv Shah, Grant Haab, Paul Petersen, and Joe Throop. Proceedings of the First European Workshop on OpenMP (EWOMP '99).

- `#pragma intel omp task`: identifica la sección de código que se ejecutará como una tarea. Permite especificar variables que se privatizan en la tarea (`private`) y que se privatizan inicializando su valor con el valor que tenian al generar la tarea (`firstprivate`).

Un ejemplo sencillo que ilustra el uso de los nuevos pragmas es el siguiente:

```
void preorder( Node& node ) {
	#pragma omp taskq
	{
		process( node.data );
		if ( node.has left )
		#pragma omp task
			preorder( node.left );
		if ( node.has right )
		#pragma omp task
			preorder( node.right );
	}
}
```

Paralelización de queens con tasking

En esta segunda opción, paralelizaremos el programa queens utilizando las extensiones a OpenMP disponibles en el compilador icc para especificar tareas.

PREGUNTAS:

P3. Modificar el programa principal para explotar el paralelismo que existe en la aplicación(en el Makefile supone que el nombre del fichero con el código fuente es `queens-task.c`). Pensad si es necesario limitar el número de niveles de taskq que se definen, y en caso afirmativo, incluid el mecanismo de control para permitirlo.

P4. Dibujar la gráfica de escalabilidad para la ejecución paralela utilizando entre 2 y 16 procesadores.