

Un vistazo al al compilador C/C++ 9.0 de Intel

CARRERA DE COMPILADORES

Intel presentó la versión 9.0 de su compilador de C++ para procesadores Intel en Junio, elevando el listón del código altamente optimizado.

POR RENÉ REBE

La versión provisional 8.1 del compilador Intel C++ Compiler (ICC) [1] introdujo el soporte para la arquitectura AMD64/x86-64 (EM64T para Intel). La versión 9 es una revisión completa llena de extensiones y optimizaciones [2]. Al igual que en versiones anteriores, el compilador soporta las arquitecturas IA-32, x86-64 e Intel Itanium. El debugger propio de Intel, una herramienta de cobertura del código y el entorno de desarrollo Eclipse completan el paquete. Un ensamblador está disponible adicionalmente para la CPU Itanium, aunque no vamos a tratar el ensamblador en este artículo. Los desarrolladores de Itanium no se han beneficiado de la integración de Eclipse hasta ahora.

El modelo de licencia es similar a la versión anterior. Existe una licencia no comercial gratuita, sin soporte, para proyectos de código abierto. Los binarios creados con esta versión no pueden ser vendidos. Para el desarrollo comercial necesitamos una licencia. En función del tamaño de la instalación, podemos especificar un número de serie o bien un archivo de licencia. El compilador puede también usar el administrador de licencias en red Flex. El compilador Intel C++ cuesta unos 300 Euros en distribuidores o unos 400 dólares si lo compramos directamente a Intel.

SSP

El Software-based Speculative Pre-Computation (SSP), activado con la

opción `-ssp`, es una característica fuertemente debatida con anterioridad al lanzamiento. Esta funcionalidad implica que el compilador añade un cierto número de hilos auxiliares al programa que pre-computan ciertas instrucciones contenidas en el mismo para rellenar la

Tabla 1: Bucles Vectorizados

ICC -O2	-O2 -ip	GCC	
Botan	(171)36	0	2
Bzip2	0	58	6
GnuPG	132	192	6
Gzip	48	62	2
Lame	118	112	22
Libmad	24	24	4
OpenSSL	148	170	30
Tramp-3D	(30)8	0	2

caché de instrucciones de la CPU con datos. Este método parece ser particularmente eficaz con el hyper-threading.

El SSP es sólo una de las llamadas optimizaciones guiadas por perfiles de ejecución (POGO). Para habilitar las POGO, el desarrollador primero tiene que compilar el programa con un instrumento especial (-prof-gen-sampling) y luego ejecutarlo con un conjunto representativo de datos de entrada, usando *profrun*. Finalmente, debe recompilar el programa completo una vez más, haciendo referencia a los datos obtenidos en los procesos previos [3].

En contraste con las versiones anteriores, -O2 o superior habilita ahora varias optimizaciones inter-procedurales, y el compilador optimiza más bucles. ICC ofrece al desarrollador incluso más feedback: los reportes referentes a bucles optimizados y otros warnings son actualmente más detallados. Así, al compilar múltiples archivos, el compilador ofrece más opciones de optimización. Adicionalmente, la opción -ipo en la nueva versión ya soporta archivos objeto individuales al compilar múltiples archivos.

Instalación con RPM

La versión 8 del Intel Compiler ocupaba unos 65 MBytes, pero la versión actual pulveriza esa cifra, al no bajar de los 192 MBytes. El tarball comprende varios paquetes RPM para I-386, EM64T y IA64. Incluye también la plataforma Eclipse completa con el C Development Toolkit (CDT) en sus versiones para GTK y Motif.

Un script se encarga de la instalación de los paquetes, pero se han reportado algunos problemas con distribuciones diferentes de Red Hat y Suse (ambas

basadas en RPM) que tienen soporte oficial de Intel. En algunos sistemas el script simplemente señala que no ha reconocido el tipo de máquina, glibc y el kernel. Por el contrario, se instaló sin ningún problema en la distribución basada en Debian, Ubuntu. Tengamos la distribución que tengamos, tendremos que usar el comando *rpm* para descomprimir el archivo RPM.

Con todo, el compilador no es tan fácil de manejar como GCC, que integra perfectamente en el sistema. ICC no mirará en los directorios de librerías y cabeceras por defecto de nuestra instalación, lo que significa que tenemos que configurar *-I/opt/intel/cc/9.0/include* y posiblemente *-I/opt/intel/cc/9.0/include/c++*, en la mayoría de los casos. Como los programas se suelen enlazar con las librerías de ICC en tiempo de ejecución, probablemente tengamos que fijar la variable de entorno *LD_LIBRARY_PATH* a */opt/intel/cc/9.0/lib* al ejecutar ICC.

Esto lo hace normalmente *source/opt/intel/cc/9.0/bin/iccvars.sh*, ya sea añadiéndolo a *.profile* o bien tecleándolo en una consola. Para hacer las cosas más fáciles, esta configuración, por defecto, puede guardarse en los archivos de configuración *.../bin/icc.cfg* y *.../bin/icpc.cfg*.

Configurando *LD_LIBRARY_PATH* guardaremos una entrada en */etc/ld.so.conf*. Por defecto, ICC usa la librería del sistema de C++, aunque proporciona su propia implementación. Las opciones *-cxxlib-icc* y *-cxxlib-gcc* soportan la conmutación explícita.

ICC versión 8.1 (o anterior) admitía las opciones del compilador GNU *-march* y *-mcpu*. Sin embargo, por algún motivo, Intel usa una notación críptica,

Listado 1: Ejemplo OpenMP

```
01 #include <omp.h>
02
03 main ()
04
05 {
06     const int N = 10000;
07     int i;
08     float a[N], b[N], c[N];
09     const int chunk = 100;
10
11     #pragma omp parallel
12         shared(a,b,c,chunk) private(i)
13     {
14
15         #pragma omp for
16             schedule(dynamic,chunk) nowait
17         for (i=0; i < N; i++)
18             c[i] = a[i] + b[i];
19     } /* Fin de bloque paralelo
    */
19 }
```

-x{K|W|N|B|P}, basada en los códigos internos de Intel. K indica Pentium III (Katmai), B para Pentium M (Banias), etc. -ax posibilita más de una CPU, permitiendo a ICC que traduzca trozos de código múltiples veces, uno para cada CPU, sin importar si sale rentable. Cuando se ejecuta el programa, los bloques optimizados se habilitan para ajustarse al tipo de procesador.

Cooperar con GCC

En principio, es posible compilar archivos objeto mezclados para un único programa usando ICC y GCC. Por ejemplo, si ICC rechaza compilar un archivo o si GCC produce código mucho mejor. Si usamos *icc* para el proceso de linkado, el linker enlazará las librerías de ejecución

Cuadro 1: OpenMP

OpenMP[4] es un estándar ampliamente reconocido que añade extensiones a los lenguajes C, C++ y Fortran para indicar explícitamente al compilador cómo distribuir los programas en hilos paralelos.

Los elementos centrales son las directivas *#pragma*, que proporcionan instrucciones de cómo debería dividir el compilador el código en fragmentos concurrentes. Por ejemplo, *#pragma omp parallel* etiqueta un bloque para la

ejecución en paralelo. *shared()* especifica variables comunes a todos los hilos, mientras que *private()* especifica variables restringidas en exclusiva a un proceso.

La directiva *#pragma omp for schedule* especifica la distribución en los hilos. Las directivas *#pragma* por tanto etiquetan un bucle para su ejecución en paralelo.

Los hilos comparten las variables *a*, *b*, *c* y *chunk*. La variable de iteración *i* es privada en cada hilo. La expresión indi-

ca al compilador que ejecute en paralelo para el bucle *for* y que divida el espacio de iteración en bloques de tamaño *chunk*.

OpenMP define más instrucciones para especificar paralelización. Sin embargo, estas instrucciones se implementan en la actualidad sólo en compiladores especiales, usualmente en aplicaciones de clustering. En Listado 1 muestra un pequeño ejemplo de cómo usar OpenMP.

de Intel necesarias sin necesidad de indicárselo explícitamente:

```
icc -o prog main.o math.o
```

Sin embargo, si queremos enlazar archivos con `gcc`, tendremos que indicar las librerías auxiliares explícitamente. GCC necesita también parámetros de enlace adicionales:

```
-L/opt/intel/cc/9.0/lib/  
-lirc
```

Tras los bastidores, el compilador de Intel para IA-32 usa las GNU Binutils, de todas formas.

Paralelización Manual y Automática

El grupo de trabajo OpenMP [4] elabora especificaciones para proporcionar ayuda a la paralelización en compiladores de C, C++ o Fortran. La opción `-openmp` indica al compilador de Intel que genere programas que paralelicen fragmentos con fuerte carga de CPU (como bucles) en sistemas Unix. Una vez se lanza el programa, el binario usa la librería Pthread para crear múltiples hilos que procesan esos fragmentos en paralelo. Así hacemos un mejor uso de los sistemas multiprocesador (véase el cuadro "OpenMP").

ICC 9 detecta segmentos de código en los cuales, hilos individuales pueden paralelizarse sin ayuda de OpenMP. La opción `-parallel` habilita la paralelización automática. Esto significa que los usuarios que quieran soporte para múltiples CPUs o múltiples núcleos, sin modificar el código fuente, verán ahora acelerados sus programas. La Tabla 1 muestra el número de bucles que vectoriza el compilador por

Listado 2: Warning ICC Detallado

```
01 lib/___dtostr.c(47): remark  
   #1572: floating-point \  
02  
03 equality and inequality compa-  
   risons are unreliable  
04  
05   if (d==0.0) {  
06  
07       ^
```

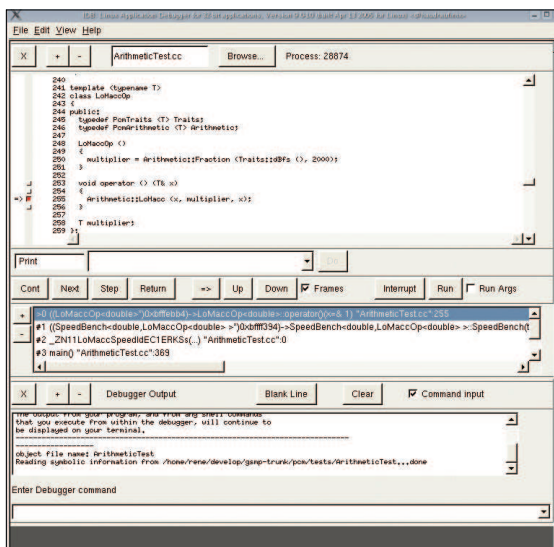


Figura 1: El front-end del compilador de Intel es funcional pero no precisamente moderno.

benchmark. Los valores entre paréntesis son las expresiones causadas por el código de la librería C++ STL.

Consejos para la Depuración

Los detallados warnings mostrados por el compilador de Intel son otra enorme ayuda para los desarrolladores. En algunos casos dicen mucho más que los warnings mostrados por el compilador GNU, especialmente en el caso de expresiones constantes, uso temporal de objetos, comparación entre números en punto flotante o pérdida de precisión de cálculos y asignaciones. Al igual que en versiones anteriores, el compilador de Intel cita el código fuente y etiqueta el carácter en el punto donde ocurrió el warning. Es una utilísima ayuda para la depuración (véase Listado 2, líneas 3 y 4).

El debugger incluido en el paquete ICC funciona con una interfaz en modo texto, al igual que su homólogo GDB, pero tiene también un pequeño front-end gráfico. El debugger IDB normalmente se ejecuta en modo DBX, en el cual espera una entrada con una sintaxis definida por Intel. Si lo ejecutamos con la opción `-gdb`, proporciona compatibilidad con GDB (versión 6.1 o superior).

Interfaz Espartana

Si ejecutamos IDB con la opción `-gui`, aparecerá una interfaz gráfica de usuario con una pinta bastante antigua (véase Figura 1). Desafortunadamente, la interfaz proporciona muy poco soporte para automatizar procesos y no simplifica casi

nada el trabajo en comparación con la consola. El debugger de la interfaz no ofrece nada en el sentido de mejor visualización.

En comparación con GDB, IDB no soporta auto-completar con el tabulador. Pero podemos fijar puntos de interrupción a la hora de crear instancias de plantillas C++, siempre que no lleven etiquetado interno por el compilador. GDB puede que acepte estos puntos de interrupción, pero no interrumpirá la ejecución del programa llegado el momento.

Conclusiones

Hemos comparado ICC 9.0 con GCC 3.4 y 4.0 compilando algunos programas de software libre (como OpenSSL, Libmad, Bzip2 y Gzip),

y hemos encontrado que las últimas versiones de GCC han reducido enormemente o eliminado las diferencias en rendimiento que una vez tuvo ICC. Nuestras pruebas no han incluido específicamente el hardware de muy alta gama o el software optimizado para cierto hardware a veces asociado con ICC. ICC 9.0 tiene aún ventaja en el caso de vectorización automática y demuestra claramente lo bien distribuido que puede estar el código compilado de C y C++ a lo largo de las unidades SIMD (Single Instruction, Multiple Data). Aspectos como el soporte para OpenMP o la paralelización automática son beneficiosos para la computación de alto rendimiento. Y los generosos warnings del compilador de Intel permiten al desarrollador detectar problemas potenciales en la fase de desarrollo, antes de llegar a la fase de depuración. ■

RECURSOS

- [1] Intel C++ Compiler <http://www.intel.com/software/products/compilers/clin>
- [2] Resumen de las optimizaciones de ICC: http://www.intel.com/software/products/compilers/docs/qr_guide.htm
- [3] Ingo A. Kubbilun, *Kernel Tuning: Compiling the Linux kernel with the Intel compiler*: http://www.linux-magazine.com/issue/45/Intel_C_Compiler.pdf
- [4] OpenMP: <http://www.openmp.org>