Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Enabling de aplicaciones

David Vicente
BSC - support

Optimizaciones

- Porting
- Compiladores
- Tuning
  - MATRIX
  - LOOP
- Librerías
  - ESSL & PESSL
  - MASS
- Herramientas
  - PROFILE
  - MPIP
  - HPMCOUNT

Pasos en el proceso de porting y optimización de aplicaciones

- Compilar y ejecutar el código (sin optimización) para verificar el funcionamiento correcto del mismo.
- Hacer un profile del código y detectar las rutinas más costosas.
- optimizar las rutinas detectadas en el profile (mejoras algorítmicas, reestructuración del código para evitar cuellos de botella , etc)
- Probar distintos niveles de optimización a nivel de compilador para mejorar el rendimiento del código
- Generar un profile del nuevo código para detectar las rutinas más costosas y instrumentarlo para poder visualizar el funcionamiento exacto del programa.
- Optimizar tanto a nivel de compilación como de código estas rutinas.
- Validar los resultados generados por el nuevo código.

Compiladores

- Uno de los puntos principales para el porting de aplicaciones es conocer los compiladores y las librerías optimizadas para la máquina donde se va a ejecutar el programa.

  - **Compiladores disponibles en MN :**

    - IBM XL compilers
      - XLC 7.0 & 8.0
      - XLF 9.1 & 10.1
    - GNU compilers
      - GCC v3.3.3
      - GCC v4.0.1
      - GCC v4.2.3

Compiladores

Table 5. IBM Compilers for PPC64 (IBM XLC-v7.0 / XLF / XLF90 v9.1) Summary

Optimizations	• -qnooptimize (Do not optimize) • -O (Optimizes generated code. Default) • -O2 (Same as -O) -O3 (Performs some memory and compile-time intensive optimizations) • -O4 (agressive optimizations) -O5 (high aggressive optimization)
32 / 64 bits	• -q32 (default) • -q64
Optimization for the PPC970-FX processor	• -qarch=ppc970 • -qtune=ppc970
VMX support	-qaltivec
OMP and threads	-qsmp=omp
Debug flags	• *-g* (produce debugging information) • *-pg* (generates extra code to write profile information suitable for gprof)

MPI	Compile with mpicc/mpiCC or mpif77/mpif90 and export the variable environment MP_CC
	`export MP_CC=xlc`
	`export MP_FC=xlf`
	`mpicc test_mpi.c -o test`
	`mpif77 test_mpi.f -o test`
	*the default object mode is 64 bits in mpi wrappers (mpicc/mpif77/mpif90)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

-q32 or –q64

- Performance consideration
  - 64-bit mode
    - Capable of handling larger amount of data directly in physical memory rather than relying on disk I/O
  - 32-bit mode
    - Smaller program, less demanding on physical memory
    - The operation of division is faster

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

-qipa

- Optimize across different files (whole program analysis)
- Have different levels

Level=0

- Program partitioning and simple interprocedural optimization

Level=1

- Default level of -qipa
- Inlining and global data mapping

level=2

- Global alias analysis
- Interprocedural data flow

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

-qinline

- Can be turned on by specifing -qipa=inline=inline-options (or -qinline=inline-options)

- Useful when your program has many subprogram calls

  Reduce the call overhead

- Identify the subprograms that are called the most and inline only those subprograms

  Examples

  - -qipa=inline=auto: inline all procedures
  - -qipa=inline=sub1:inline=noauto: only inline the procedure sub1

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

-OX

- -On option
  - -O0: very limited optimization, fast compilation, debuggable code
  - -O2: comprehensive low-level optimization, partial debug support
  - -O3: more extensive optimization, some precision trade-off
  - -O4: Everything from -O3 plus -qhot -qipa -qarch=auto -qtune=auto -qcache=auto
  - -O5: Everything from -O4 plus -qipa=level=2

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tuning

- Lo recomendable a la hora de portar para MN es :
  - Usar los compiladores de IBM
  - Usar las librerias MASS, ESSL y PESSL (detalles más adelante)
  - Optimizar para máquinas ppc970
  - Intentar no romper el código en gran numero de rutinas para evitar el overhead que puede llegar a producir este tipo de llamadas.

- Por otro lado, los compiladores a veces no son capaces de hacer todas las optimizaciones posibles, por lo tanto es importante hacer algunas optimizaciones a mano para facilitarles el trabajo.

Tuning

Array Allocation in C and Fortran

Example 4 by 4 matrix with data values

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

C: Row-Major Order: rows of the matrix are stored contiguously

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Fortran: Column-Major Order: columns of the matrix are stored contiguously

1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

```
DO J=1,1000
  DO I=1,1000
     C(I,J) = A(I,J) * B(I,J)
  END DO
END DO
```

```
for(i=0;i<1000;i++)
  for(j=0;j<1000;j++)
     c[i][j] = a[i][j] * b[i][j]
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tuning

- Fusión de bucles :
  - Unir varios bucles igualmente indexados en uno solo
- Ventajas
  - El overhead del bucle se reduce
  - Mejora el overlap de instrucciones
  - Si los bucles hacen referencias comunes se puede reducir el numero de errores de cache.

```
for(i=0;i<100000;i++)
        x = x * a[i] + b[i];
for(i=0;i<100000;i++)
        y = y * a[i] + c[i];
```

→

```
for(i=0;i<100000;i++)
  x = x * a[i] + b[i];
  y = y * a[i] + c[i];
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ESSL

- Tiene más de 400 rutinas
- Tuneado para PowerPC systems
- Disponible en paralelo con Myrinet (pessl)
- Uso
  - Linkar con -lessl

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ESSL

- Linear algebra subprograms
- Linear equations
- Eigen system analysis
- Fourier transforms
- Convolution and correlation
- Sorting and searching
- Interpolation
- Numerical quadrature
- Random number generation

MASS

- Intrinsic vector functions
  - Los compiladores de IBM intentan generar Intrinsic vector functions con la opción -qhot
  - Ejemplos: vlog, vexp, vdiv, vsqrt
  - Estas librerías estan muy optimizadas

```
Do i=1,n
   A(i)=log(B(i))
Enddo
```

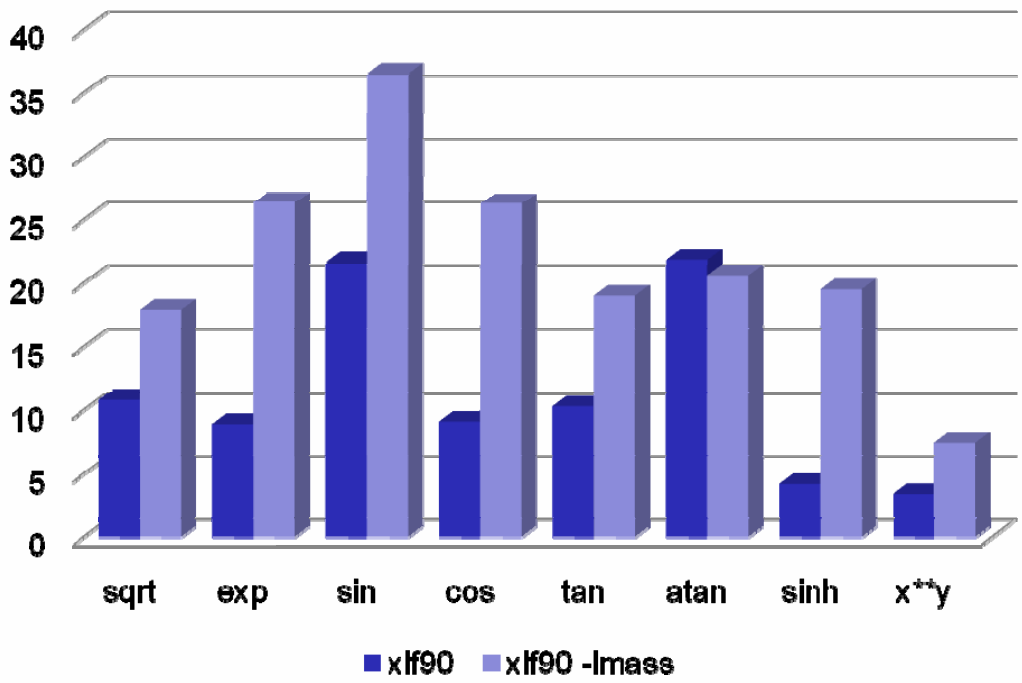
-qhot →

```
Call __vlog(A,B,n)
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MASS

- Uso
  - Se pueden llamar de forma explicita :
    - Ejemplo: call vexp(A,B,n) en vez de do i=1,n; B(i)=exp(A(i)); enddo;
  - Al linkar es necesario incluir –lmassv



Herramientas

- PROFILE:
  - Compilar el código con el flag –pg (soportado tanto en GCC como en los compiladores de IBM)
  - Al ejecutar el binario se generara un fichero gmon.out que contiene toda la información de profile
  - Para visualizar el contendido del fichero hay que ejecutar :
    - gprof “binario”
  - Para binarios MPI se recomienda usar el siguiente script para evitar que el fichero gmon.out sea sobrescrito por todos los procesos :

```
#!/bin/bash
export GMON_OUT_PREFIX=$1$GMPI_ID
$@
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Herramientas

- MPIP:
  - Permite generar un resumen de las comunicaciones MPI del programa, su incidencia en el rendimiento es menor que otras herramientas de instrumentación.
  - Uso :
    - No hace falta recompilar, simplemente se tiene que ejecutar el programa de la siguiente manera :

```
>srun /gpfs/apps/MPIP/3.0.1/64/bin/mpip-exe ./binario64
```

O

```
>srun /gpfs/apps/MPIP/3.0.1/32/bin/mpip-exe ./binario32
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Herramientas

- HPMCOUNT:
  - Genera un grupo de estadísticas mediante contadores PAPI, el overhead que introduce en la ejecución del código es mínimo
  - Uso :

```
source /gpfs/apps/IHPCT/2.1/env_sh

hpmcount ./binario
```


Barcelona Supercomputing Center
Centro Nacional de Supercomputación