#SHAREorg

SHARE
Technology · Connections · Results

11669: Make Your C/C++ and PL/I Code FLY With the Right Compiler Options

Visda Vokhshoori

IBM

Peter Elderon

IBM

SHARE in Anaheim 2012

AGENDA

- What does application performance mean to you and how do you achieve it?
- Compiler knobs
- Summary

What does application performance mean to you and how do you achieve it?

- Reduction of code size
- Reduction in the time spent running
- Reduction in the time spent compiling

- Newer hardware
- Code for performance
- Utilize compiler option

-Often design is around a functionality of a feature

-Tuning the application for better performance is _often_ (not always) an after thought and in reaction to an observed non-ideal transaction response time or througput

-The performance of the transaction is subject to change

In our view Performance means:

Optimizing your enterprise's IT investment and IT productivity

ARCH

-The ARCH option specifies the architecture for which the executable program's instructions are to be generated.

-Default is ARCH(5) produces code available on 2064-xxx (z900)

Architecture

	Machine supported	Hardware features exploited
ARCH(5)	z900	64-bit support,…
ARCH(6)	z990	Support long displacement facility,…
ARCH(7)	z9	Extended immediate support, Decimal Floating Point,…
ARCH(8)	z10	Compare and Branch Add Logical with Signed Immediate, …
ARCH(9)	z196	Conditional Load/Store, Non-desctructive ops, high word,…

ARCH 7

PL/I

- >= 7: srstu used to inline some widechar index
- >= 7: trtr used to inline some searchr and verifyr
- >= 7: cu12 used to inline some ulength and uvalid
- >= 7: cu21 used to inline some ulength and uvalid
- >= 7: cu12 used to inline memcu12
- >= 7: cu21 used to inline memcu21

C/C++ & PL/I

- On z9 machine IEEE DFP was added in software, millicode
- On z10 machine support added to the hardware
- Use of –qdfp –qarch=7 to enable the new dfp instructions
- In addition decimal types can be used to facilitate DFP programming
- In the experiments using DFP provided 5x speedup

ARCH 8

PL/I

>= 8: used to inline conversions between hex/ieee float and dfp float

- ADD LOGICAL WITH SIGNED IMMEDIATE
- COMPARE AND BRANCH (RELATIVE)
- COMPARE (HALFWORD) RELATIVE LONG
- COMPARE IMMEDIATE AND BRANCH (RELATIVE)
- COMPARE LOGICAL AND BRANCH (RELATIVE)
- COMPARE LOGICAL IMMEDIATE AND BRANCH (RELATIVE)
- COMPARE LOGICAL RELATIVE LONG
- LOAD HALFWORD RELATIVE LONG
- LOAD LOGICAL (HALFWORD) RELATIVE LONG
- LOAD RELATIVE LONG
- MULTIPLY SINGLE IMMEDIATE
- STORE (HALFWORD) RELATIVE LONG
- and also the PFPO instruction (useful in converting bewteen DFP and hex or between DFP and IEEE binary)

C/C++

- Compare and Branch
- Prefetch
- ADD LOGICAL WITH SIGNED IMMEDIATE
- COMPARE AND BRANCH (RELATIVE)
- COMPARE (HALFWORD) RELATIVE LONG
- COMPARE IMMEDIATE AND BRANCH (RELATIVE)
- COMPARE LOGICAL AND BRANCH (RELATIVE)
- COMPARE LOGICAL IMMEDIATE AND BRANCH (RELATIVE)
- COMPARE LOGICAL RELATIVE LONG
- LOAD HALFWORD RELATIVE LONG
- LOAD LOGICAL (HALFWORD) RELATIVE LONG
- LOAD RELATIVE LONG
- MULTIPLY SINGLE IMMEDIATE
- STORE (HALFWORD) RELATIVE LONG

ARCH 8 (C/C++)

-Data prefetch instruction PFD
  Because of memory latencies, cache misses can be very expensive
  Tell h/w to pull data from memory into cache
  Have seen 60% speedup with it's use (program did a lot of data movement)
  IBM JVM uses it for double-digit improvement of GC
  Compiler can insert prefetch instructions for you (minimum: ARCH=8 and at least HOT)
  Compiler currently only recognizes loop-based traversal over arrays
  Will not catch things like red-black trees etc
-XL z/OS C/C++ compiler provides built-ins, a C-level routine to use directly
  __dcbt, __dcbst, __dcbf and __dcbtst (same as on AIX)
http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.r12.cbcpx01%2Fcbcpg1b0334.htm
-Exploiting the z10 prefetch built-in
  example:
https://www.ibm.com/developerworks/mydeveloperworks/blogs/5894415f-be62-4bc0-81c5-3956e82276f3/entry/exploiting_the_z10_prefetch_built_in25?lang=en
-Biggest problem is knowing where to use it, and tuning it
  Profiler & developer knowledge can help!

ARCH 9

PL/I and C/C++

- High-word facility
- Interlocked-access facility
- Load/store-on-condition facility
- Distinct-operands facility
- Population-count facility

ARCH9: Load/store on condition facility

consider this small program:

```
2.0 | test: proc returns( fixed bin(31) );
3.0 |
4.0 | exec sql include sqlca;
5.0 |
6.0 | dcl c fixed bin(31);
7.0 |
8.0 | exec sql commit;
9.0 |
10.0 | if sqlcode = 0 then
11.0 | c = 0;
12.0 | else
13.0 | c = -1;
14.0 |
15.0 | return( c );
16.0 | end;
```

- Under OPT(3) ARCH(8), the instructions after the call are:

```
0000CA  0DEF            000008 |              BASR    r14,r15
0000CC  5800  D0F4      000010 |              L       r0,<a1:d244:l4>(,r13,244)
0000D0  A718  FFFF      000010 |              LHI     r1,H'-1'
0000D4  EC06  0005 007E 000010 |              CIJNE   r0,H'0',@1L8
0000DA  4110  0000      000010 |              LA      r1,0
0000DE                  000010 |       @1L8  DS      0H
0000DE  58E0  2000      000015 |              L       r14,_addrReturns_Value(,r2,0)
0000E2  5010  E000      000015 |              ST      r1,_shadow1(,r14,0)
```

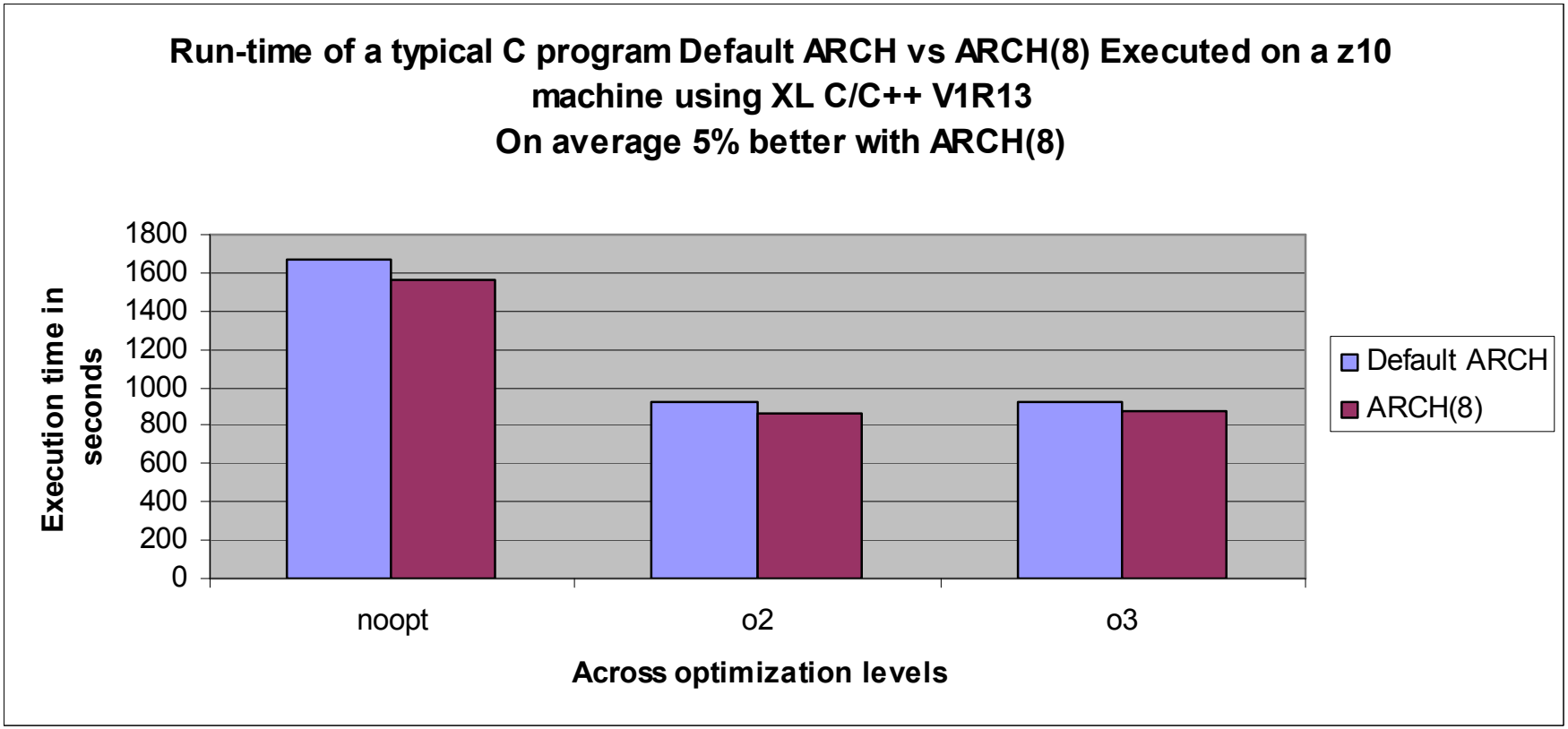
- But, under OPT(3) ARCH(9), the instructions after the call are:

```
0000CA  0DEF          000008 |        BASR    r14,r15
0000CC  A718  FFFF    000010 |        LHI     r1,H'-1'
0000D0  BF0F  D0F4    000010 |        ICM     r0,b'1111',<a1:d244:l4>(r13,244)
0000D4  58E0  2000    000015 |        L       r14,_addrReturns_Value(,r2,0)
0000D8  4100  0000    000010 |        LA      r0,0
0000DC  B9F2  8010    000010 |        LOCRE   r1,r0
0000E0  5010  E000    000015 |        ST      r1,_shadow1(,r14,0)
```

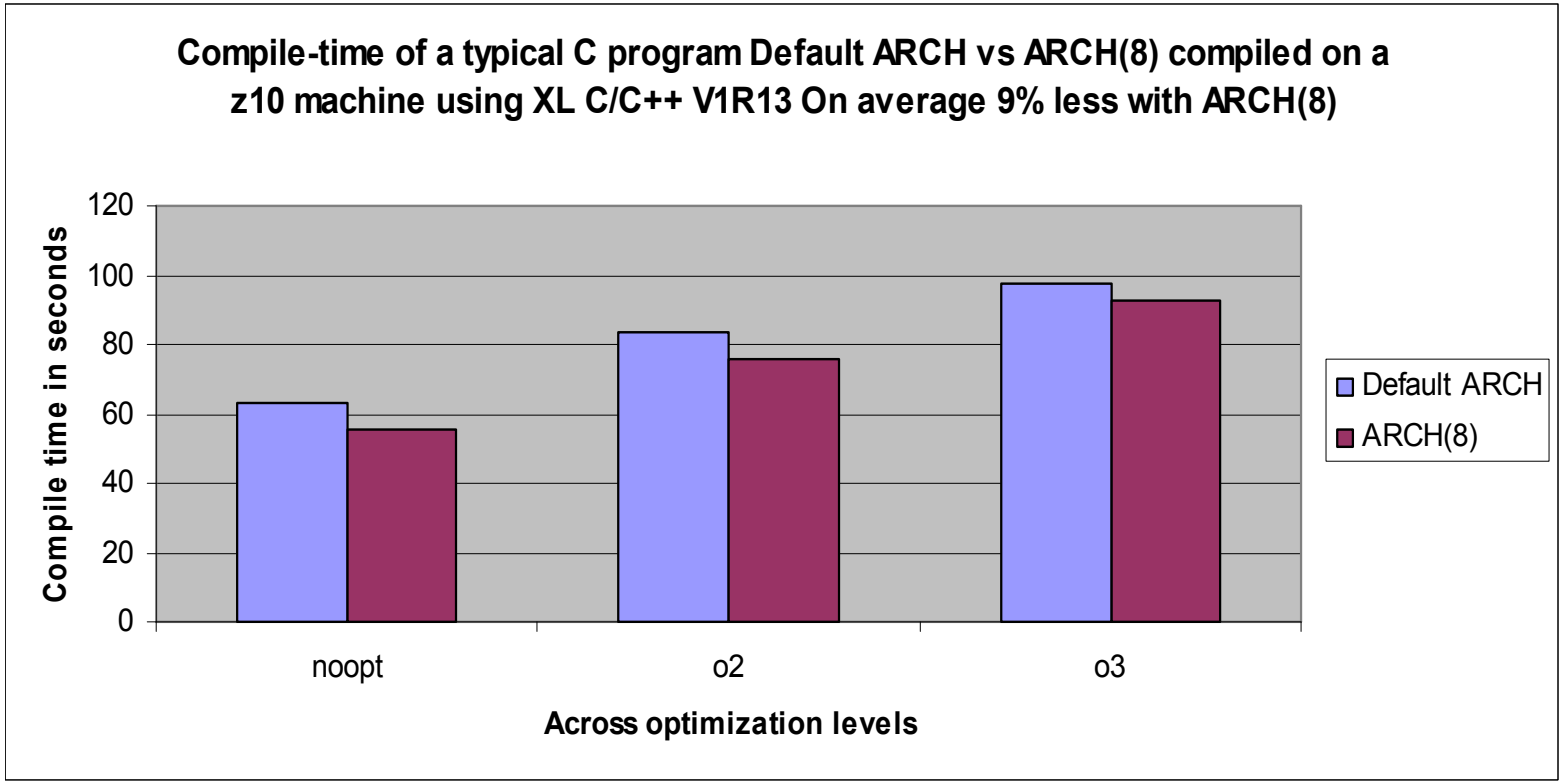
- So, under ARCH(8), the code sequence was:Load SQLCODE into r0
- Load -1 into r1
- Compare r0 (SQLCODE) with 0 and branch if NE to @1L8
- Load 0 into r1
- @1L8
- Store r1 into the return value

- While under ARCH(9), the code sequence has no label and no branch:Load -1 into r1
- Load SQLCODE into r0 via ICM (so that CC is set)
- Load 0 into r0
- Load-on-condition r1 with r0 if the CC is zero (i.e. if SQLCODE = 0)
- Store r1 into the return value

Benefit of specifying ARCH option



ARCH option does NOT increase compile time

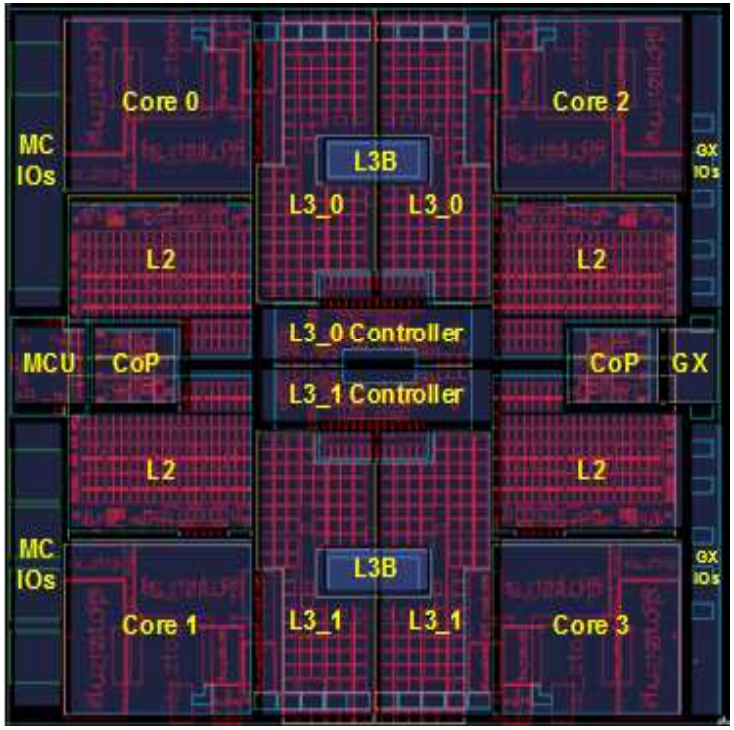


TUNE* (C/C++)

-Tunes instruction selection, and other implementation-dependent performance enhancement for a specific implementation of a hardware architecture

-Default is TUNE(5) produces code available on 2064-xxx (z900)

-TUNE impacts with OPTIMIZATION



Recommendation for Architecture Setting

Choosing the right hardware architecture target is important especially when application is known to run on a specific type of machine.

For C/C++ application we recommend the TUNE option to match the ARCH option or be greater. TUNE is primarily an optimization option

Help the compiler understand your code

-Value add of compilers realized when optimization is turned on

-The un-optimized, default follows all the source code lines literally; lots of loads and stores

-With optimization compiler will keep the intermediate values in registers avoiding memory updates until the final value is calculated

-Address calculations inside loops are optimized such that as much as possible is done outside of the loop

-Optimization is inhibited by non-ANSI aliasing source code, e.g. unsafe pointer casting

```
*(unsigned long long *)(buf + 8) = (unsigned long long)code;
```

-When –qalias=noansi compile option is specified optimizer makes pessimistic aliasing assumptions such as all pointers can point to any object whose address is taken, regardless of type. Optimizing such source code may cause more harm than benefit, especially if it is a performance critical part of the application

-The amount of improvement varies depending on the specific application

-Increasing the optimization level will significantly increase compile time

Summary of key optimization options

Optimization Level xlc: -O batch/PL/I: OPT()	C/C++	PL/I
2	Implies NOANSIALIAS inlining, instruction scheduling, unrolling, dead code removal, common sub-expression elimination, more dead code,…	Same as C/C++ except individual optimizations run less # of times, dead code, or less # of individual optimization are run, CSE
3	Implies ANSIALIAS, all above plus instruction level parallelism, more pattern matching,NOSTRICT	Identical to C/C++ OPT(2)

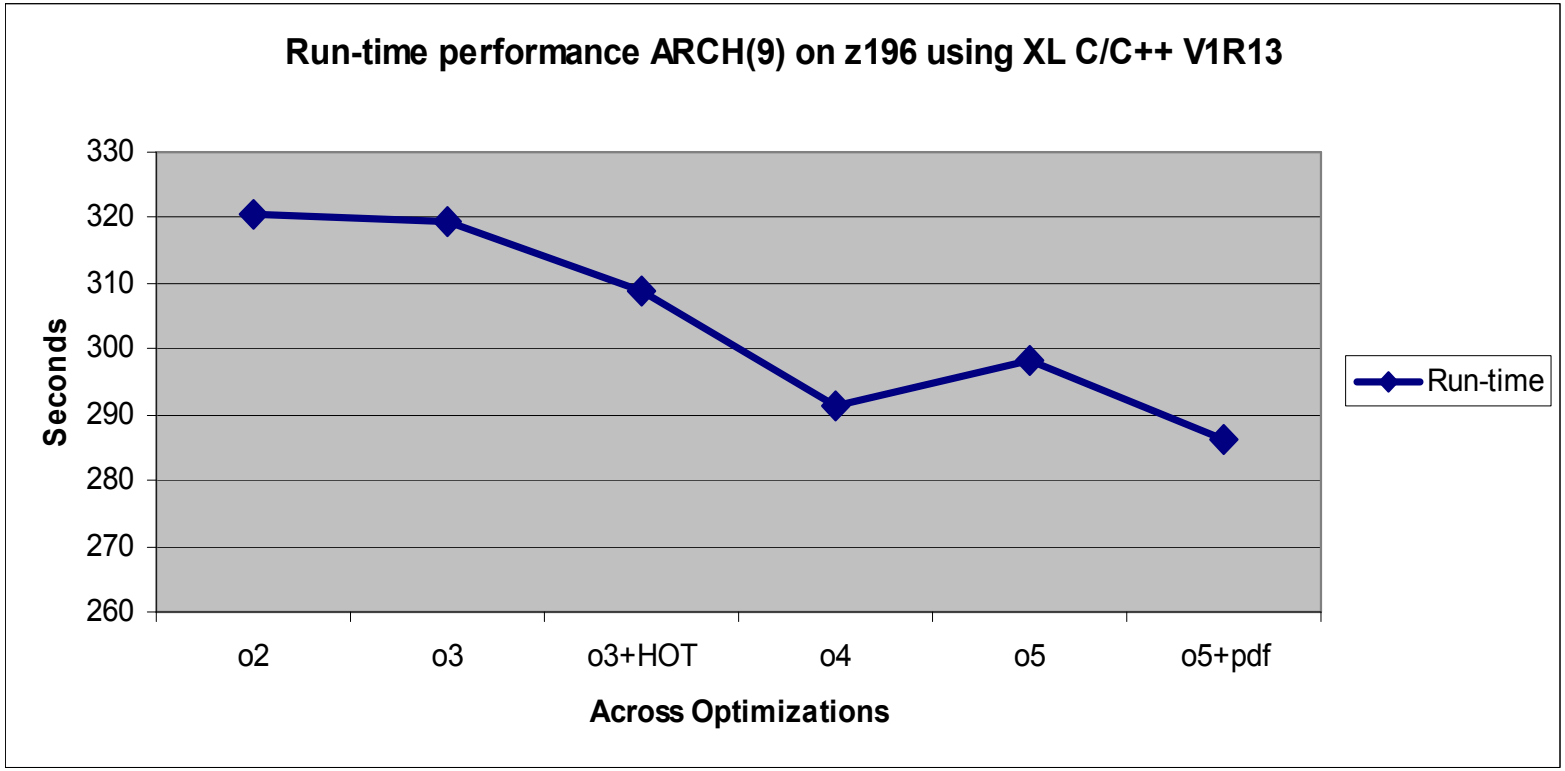
Summary of key optimization options

Optimization Level xlc: -O batch: OPT()	C/C++	PL/I
4	Optimization decision made for the sub-program: high order transformation of loops, pointer analysis, constant propagation, …	N/A
5	Optimization decision made for the entire program which could lead to significant improvement in the generated code at cost of significant compile time	N/A

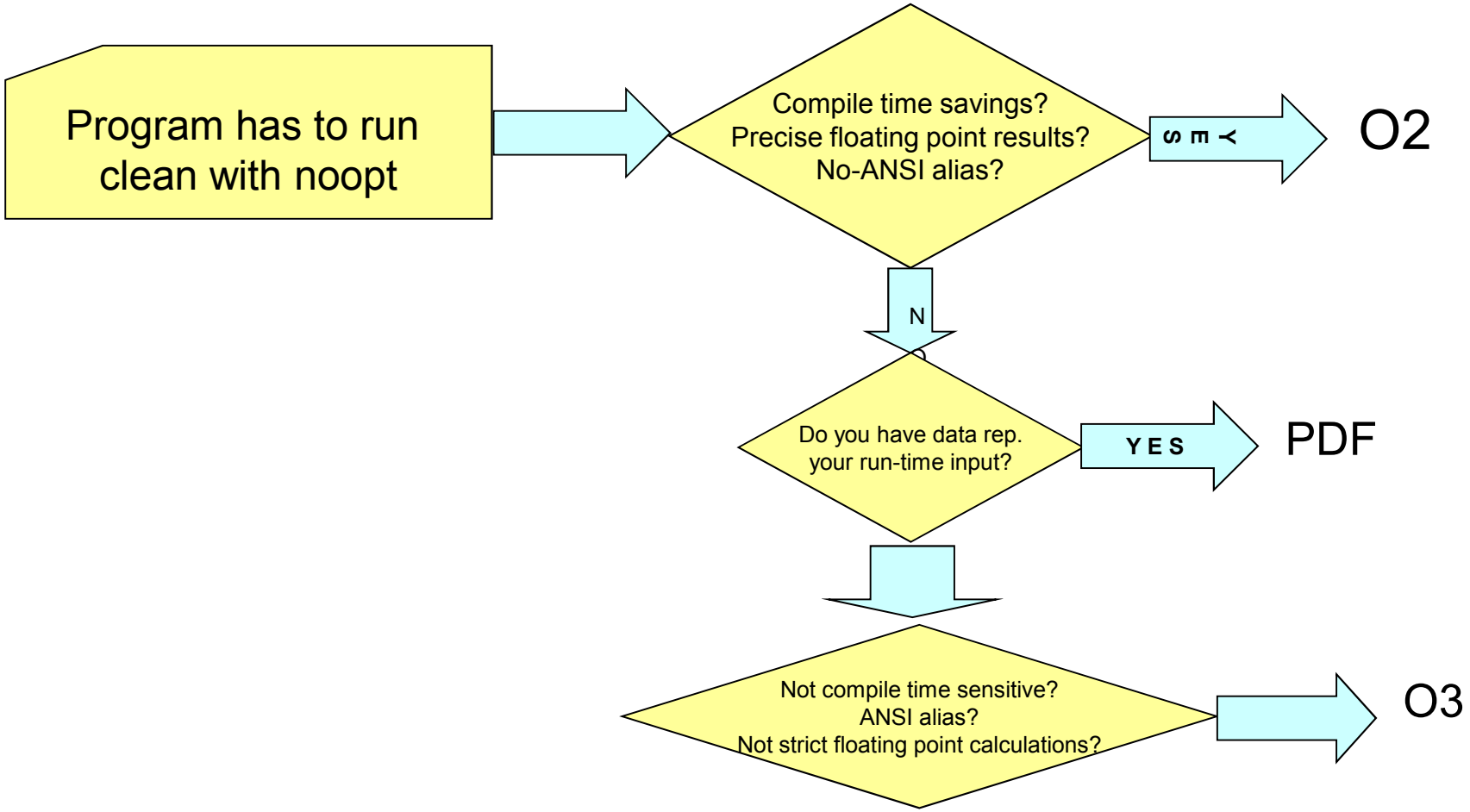
Optimization options (C/C++)

Optimization Level xlc: -q batch:	C/C++	PL/I
HOT	At least requires OPT(2). Exploiting parallelism in loops. Loop fusion, loop distribution, loop parallelization	N/A
PDF1/PDF2	Requires at least OPT(2). Requires training data. With PDF can do loop alignment on cache line boundary	N/A

Improvement in run-time performance with higher optimization



Recommendation for compiling with optimization levels



C/C++ compiler option to help performance Inlining

-Which one of the following yields a better execution time?

a) –O2 –qinline

b) –O2 –qnoinline

c) requires tuning

-A C program could perform 4.5X worse with INLINE

-Using the inline sub-option is one way to help the right balance

-#pragma inline* is another way to hint the compiler as to which functions to inline

*C only

Example of a C program using inline sub-options

Execution Time in seconds
-O2 -qnoinline 25.71

-qinline=auto:report:_threshold_:1000

threshold	execution time in seconds	
:100	111.98	*default
:150	111.97	
:190	10.72	
:200	10.72	
:250	10.73	
:300	10.78	

Table 1. Shows the execution time with respect to different inlining thresholds

XPLINK

-A modern linkage convention that is 2.5 times more efficient than conventional

-Have seen some programs improve by 30%

-Cannot statically link non-XPLINK with XPLINK, so your hands may be tied

-Can call non-XPLINK DLLs from XPLINK DLLs and vice-versa but must tell compiler about this so it generates translation code. These calls are more expensive

-If XPLINK to XPLINK calls more frequent than noXPLINK to XPLINK, then it's a win, otherwise, a loss

*http://www.redbooks.ibm.com/abstracts/sg245991.html

Feedback

- I am collecting feedback for future sessions, which topics are you interested in?
- Looking forward to hearing from you!!