Wigner GPU Lab

Tale of the Programming Language Wanderer

Migrating from Fortran and C to modern C++

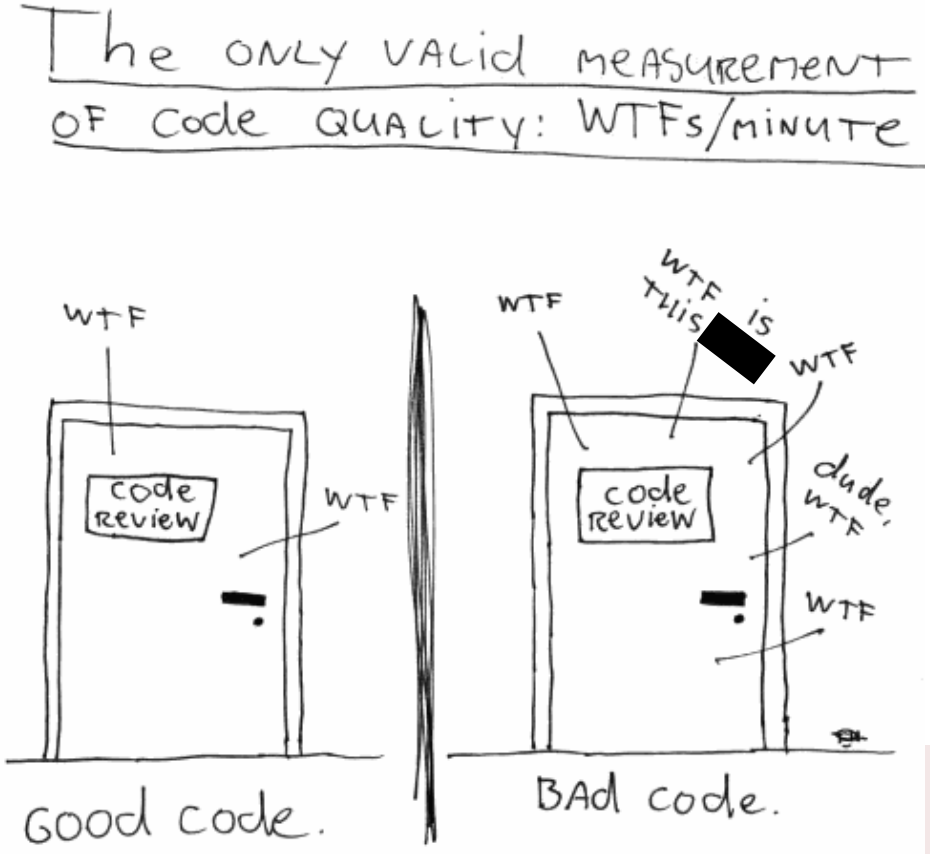
Lectures on Modern Scientific Programming
Wigner RCP
23-25 November 2015

Fortran

D. Berényi – M. F. Nagy-Egri – Sz. Harangozó

The journey

- Motivation
  - FORmula TRANslator codes usually written in older versions (77 - 90)
    Nobody likes software archeology…
  - Subprograms may have newer versions (HIJING - Pythia)
- Planning
  - Understanding original structure
  - If necessary creating a new one
- Execution
  - Patience … … …
- Prize
  - To have an object oriented, understandable, fast code



Inventory of the Wanderer I.

- Plates and Shield
  - Tutorials

    cause one can not be expert in FORTRAN
  - Forums

    „yeah, I also met with this problem”
  - Professor

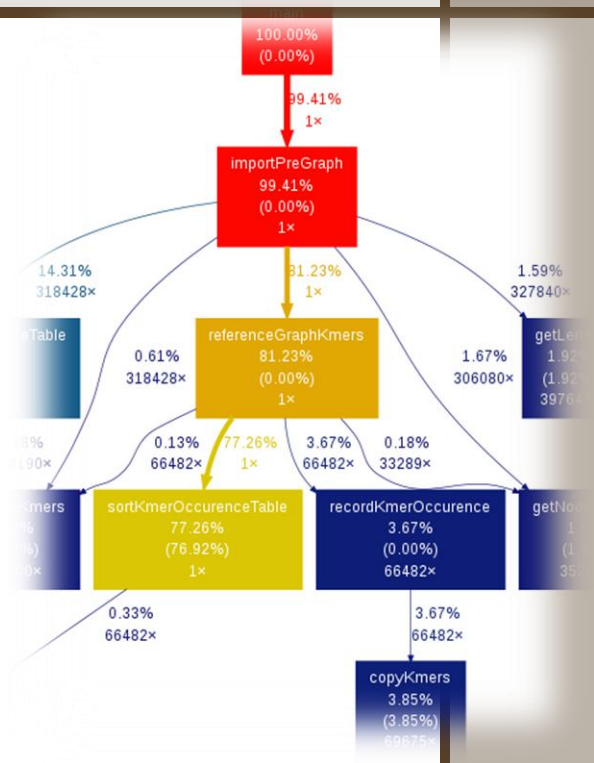
    actually understands what’s going on

- Weapons
  - GNU profiler

    dot helps you see
  - STL, GSL

    already done
  - Professor

    actually understands what’s going on

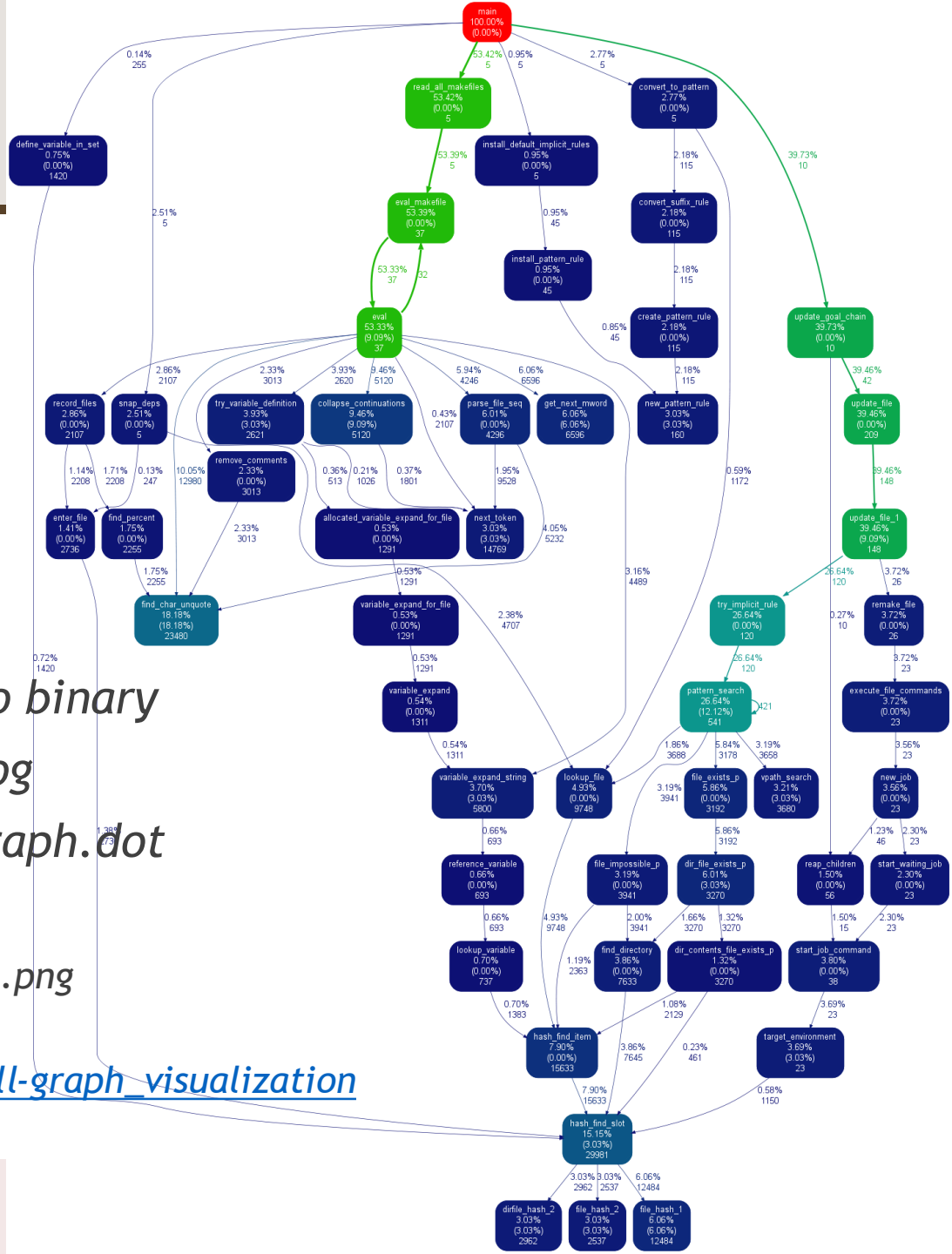


Fortran

Inventory II. the profiler

- Reqirements:
  - Python 3.x
  - Xdot *sudo apt-get install xdot*
  - Gprof2Dot *sudo apt-get install graphviz*
- Usage:
  - compile with **-pg** flag *gcc -pg -g source.cpp -o binary*
  - run code with **gprof** *gprof ./binary > out.log*
  - generate callgraph *gprof2dot out.log > graph.dot*
  - view result *xdot graph.dot*
  - optionally convert to picture *dot -Tpng graph.dot -o graph.png*

https://redmine.epfl.ch/projects/python_cookbook/wiki/Gprof_call-graph_visualization



Monsters or Beasts I.

- Datatypes and Arrays

```
      INTEGER   I,J,NI,NJ
      PARAMETER(NI=2,NJ=3)
      INTEGER   B(NI,NJ)                same as B(2,3)

      DO J = 1, NJ                      loop to 3
          DO I = 1, NI                  loop to 2
              PRINT 10, I, J, B(I,J)
10            FORMAT('B(', I1, ',', I1, ') =', I2)
          END DO
      END DO
```

```
int a[3][2];

for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 2; j++) {
        printf("a[%d][%d] = %d\n", i, j, a[i][j]);
    }
}
```

http://www.yolinux.com/TUTORIALS/LinuxTutorialMixingFortranAndC.html

FORTRAN	C/C++
byte	unsigned char
integer*2	short int
integer	long int or int
integer iabc(2,3)	int iabc[3][2];
logical	long int or int
logical*1	Bool (C++, One byte)
real	float
real*8	double
real*16	long double
complex	struct{float realnum; float imagnum;}
double complex	struct{double dr; double di;}
character*6 abc	char abc[6];
character*6 abc(4)	char abc[4][6];
parameter	#define PARAMETER value
counts from 1	counts from 0
column-major	row-major dimension order

Monsters or Beasts II.

- Functions
  - 90

```
function func(i) result(j)
   integer, intent(in) :: i ! input
   integer             :: j ! output
   j = i**2 + i**3
end function func
```

```
int func(int i) {
    return i*i + i*i*i;
}
```

  - 77

```
FUNCTION func_name(a, b)
   INTEGER :: func_name
   INTEGER :: a
   REAL    :: b
   func_name = (2*a)+b
   RETURN
END FUNCTION
```

```
int func_name(int a, float b) {
    return 2 * a + (int)b;
}
```

- Subroutines

```
subroutine square_cube(i,isquare,icube)
  integer, intent(in)  :: i              ! input
  integer, intent(out) :: isquare,icube ! output
  isquare = i**2
  icube   = i**3
end subroutine square_cube
```

intent(in/out/inout) attribute

```
void square_cube(int i, int& isquare, int& icube) {
    isquare = i*i;
    icube = i*i*i;
}
```

→ **NO CHECK!**

Fortran

Enemy champions

- COMMON BLOCKs
  - global variables
  - „extern struct”

```
DOUBLE PRECISION X
INTEGER A, B, C
COMMON/ABC/ X, A, B, C
```

→

```
extern "C" {
    extern struct abc_ {
        double x;
        int a, b, c;
    };
}
```

- They are guiding us!

```
FUNCTION AREA(DUMMY)
COMMON/RECTANGLE/ A, B
AREA=A*B
RETURN
END FUNCTION

FUNCTION PERIMETER(DUMMY)
COMMON/RECTANGLE/ A, B
PERIMETER=2*(A+B)
RETURN
END FUNCTION
```

→

```
class Rectangle {
    int width, height;
public:
    Rectangle(int, int);
    int area() { return (width*height); }
    int perimeter() { return 2 * (width*height); }
};

Rectangle::Rectangle(int a, int b) {
    width = a;
    height = b;
}
```

Fortran

Arch Enemy

- GOTO statements
  - Too many possibilities
    - do – while
    - continue
    - break
    - loop entry point
    - exception handling
  - Unreadable
  - still exists in C++

    rethinking is
    - time-consuming
    - not failproof

    but still worth it…

```
      FUNCTION NGCD(NA, NB)
      IA = NA
      IB = NB
1     IF (IB.NE.0) THEN
        ITEMP = IA
        IA = IB
        IB = MOD(ITEMP, IB)
        GOTO 1
      END IF
      NGCD = IA
      RETURN
      END
```

```
int ngcd(int na, int nb) {
    int itemp;
    while (nb != 0) {
        itemp = na;
        na = nb;
        nb = itemp%nb;
    }
    return na;
}
```

The Right Way

```
int ngcd(int na, int nb) {
    int itemp;
    L1:
    if (nb != 0) {
        itemp = na;
        na = nb;
        nb = itemp%nb;
        goto L1;
    }
    return na;
}
```

The Bad Way

Arch Enemy

- GOTO … hell

```
      N = 100
      DO 10 I = 1, N
         DO 10 J = 1, N
         ...
         IF(J.GE.15) GOTO 10
         ...
         IF(I.EQ.54) GOTO 20
         ...
15
         ...
10       CONTINUE
...
20       CONTINUE
...
      IF(...) GOTO 15
```

define a function for that

```
bool double_loop(int N,bool entry) {
    bool exit = false;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (entry) {
                //...
                if (j >= 14) continue;
                //...
                if (i == 53) {
                    exit = true;
                    break;
                }
                //...
            }
            entry = true;
            //...
        }
        if (exit) return false;
    }
    return true;
}
```

in the main

```
bool entry = true;
do {
    //... between 10 and 20
    if (double_loop(N,entry)) {
        //... between 20 and goto 15
    }
    entry = false;
} while (something);
```

Fortran

Traitor

- IMPLICIT statements
  - confirms or changes the default type of names
  - Eg.: IMPLICIT REAL*8(A-H,O-Z)

  (all variables beginning with any of the above letters are by deafult floating numbers)

  - Yeah, no need for declaration!
    - Typos → compiling without errors
    - undefined variables → compiling without warnings

  - Already accepted codes?

```
IMPLICIT INTEGER (A-Z)
IMPLICIT REAL (A-C)
C = 1.5E8
D = 9
```

```
float c = 1.5e8;
int d   = 9;
```

Luckily no way to do that…

Using what we already have I.

Why is this slide here?

Standard Template Library

- std::vector

A sequence container that encapsulates dynamic size arrays.

Specialized space-efficient dynamic bitset: *vector<bool>*

- Random

```
// Seed with a real random value, if available
std::random_device rd;
// Choose a random mean between 1 and 6
std::default_random_engine e1(rd());
std::uniform_int_distribution<int> uniform_dist(1, 6);
int mean = uniform_dist(e1);
// Generate a normal distribution around that mean
std::mt19937 e2(rd());
std::normal_distribution<> normal_dist(mean, 2);
```

- Algorithms

```
// sort using a custom function object
struct {
    bool operator()(int a, int b) { return a < b; }
} customLess;
std::sort(s.begin(), s.end(), customLess);
```

Fortran

```
call bubble_sort(Item_1, N1)
call bubble_sort(Item_2, N2)
call bubble_sort(Item_7, N7)
call bubble_sort(Item_8, N8)

subroutine bubble_sort(b,n)
implicit none
integer n, i,j, b(9000),itemp
real a(n), temp
real  Crn_rapdt(9000), Crn_mt0(9000)
common/bb_sort/Crn_rapdt,Crn_mt0
do i=1,n
  a(i)=Crn_mt0(b(I))
enddo
do i=n-1, 1, -1
  do j=1,i
    if ( a(j) < a(j+1) ) then
      temp=a(j)
      a(j)=a(j+1)
      a(j+1)=temp

      itemp=b(j)
      b(j)=b(j+1)
      b(j+1)=itemp
    end if
  end do
end do
return
end subroutine
```

Using what we already have II.

- GNU Scientific Library

  http://www.gnu.org/software/gsl/

  Problem: gsl_function class != std::functional

```
struct gsl_function_struct
{
  double (* function) (double x, void * params);
  void * params;
};
typedef struct gsl_function_struct gsl_function ;
```

```
template< typename F >
class gsl_function_pp : public gsl_function {
public:
    gsl_function_pp(const F& func) : _func(func) {
        function = &gsl_function_pp::invoke;
        params = this;
    }
private:
    const F& _func;
    static double invoke(double x, void *params) {
        return static_cast<gsl_function_pp*>(params)->_func(x);
    }
};
```

Use templates instead of std::functionals as a wrapper class member, cause the compiler usually has an easier time optimizing for that

Usage:

```
Class* ptr2 = this;
auto ptr = [=](double x)->double {return ptr2->foo(x); };
gsl_function_pp<decltype(ptr)> Fp(ptr);
gsl_function *F = static_cast<gsl_function*>(&Fp);
```

http://stackoverflow.com/

Additional difficulties

- Compiler options
  - *-fno-automatic*

  Treat each program unit (except those marked as RECURSIVE) as if the SAVE statement were specified for every local variable and array referenced in it.

  - *-frecursive*

  Allow indirect recursion by forcing all local arrays to be allocated on the stack.

  - *-falign-commons*

  By default, gfortran enforces proper alignment of all variables in a COMMON block by padding them as needed. **PITFALL!**

  https://gcc.gnu.org/onlinedocs/gfortran/Code-Gen-Options.html

Additional difficulties - solution

- Compiler options
  - *-fno-automatic*

SAVE statement forces every function start form it's last state.
One can simply create the function variables as *Class members*.

  - *-frecursive*

One can forward declare a template class.

  - *-falign-commons*

This is only a problem if you want to keep some FORTRAN code parts.
Could lead to data corruption, but one can simply just forgot it.

Additional difficulties - solution

- Compiler options
  - *-fno-automatic*

```
      FUNCTION ROMG(X)
      DIMENSION FR(0:1000)
      DATA I0/0/
      IF(I0.NE.0) GO TO 100
      DO 50 I=1,1001
      XR=(I-1)*0.01
      FR(I-1)=OMG0(XR)
50    CONTINUE
100   I0=1
      IF(X.GE.10.0) THEN
            ROMG=0.0
            RETURN
      ENDIF
      IX=INT(X*100)
      ROMG=(FR(IX)*((IX+1)*0.01-X)+FR(IX+1)*(X-IX*0.01))/0.01
      RETURN
      END
```

```
double HiRand::romg(double x){
    double xr;
    if(i0eik == 0){
        fr[0] = 0.0;
        for(int i=1; i<=1001; i++){
            xr = (i-1)*0.01;
            fr[i-1]=omg0(xr);
        }
        i0eik = 1;
    }
    if(x >= 10) return 0.0;
    int ix = (int)(x*100);
    return (fr[ix]*((ix+1)*0.01-x)+fr[ix+1]*(x-ix*0.01))/0.01;
}
```

Source file

See the cheating?

Header file

```
private:

    // Eikonal table
    double fr[1001];
    int i0eik;
```

Fortran

Alliance – Using Fortran and C++ together

- Call Fortran in C++
  - Fortran – nothing special

```
subroutine fortfunc(ii,ff)
integer ii
real*4  ff
...
return
end
```

  - C++ - create extern

```
extern"C" {
    void fortfunc_(int *ii, float *ff);
}

fortfunc_(&ii, &ff);
```

  - Compile – link Fortran

```
gfortran - c testF.f
g++ - c testC.cpp
g++ - o test testF.o testC.o - lgfortran
```

- Call C++ in Fortran
  - Fortran – call without underscore

```
PROGRAM main
IMPLICIT NONE
INTEGER::ii
REAL::ff
CALL cpp_func(ii, ff);
END PROGRAM
```

  - C++ - create extern

```
extern "C" {
    void cpp_func_(int *ii, float *ff) {
        cpp_func(*ii, *ff);
    }
}
```

  - Compile – link C++

```
gfortran - c testF.f
g++ - c testC.cpp
gfortran - o test testF.o testC.o - lstc++
```

The Loot

- Finding still unknown errors
- Paralellization opportunities
- Not just the Professor actually understands what's going on

„We always have the code, and the code will guide us."

Have a pleasant journey!