

# Using OpenIMPACT

Robert Kidd

Gelato Meeting

Tsinghua University, Beijing

October 12, 2004

- ⊕ OpenIMPACT is now available for download!
- ⊕ We now invite the entire Gelato community to try out OpenIMPACT before its wider release

<http://gelato.uiuc.edu>

# http://gelato.uiuc.edu



## Visit gelato.uiuc.edu

- Downloads
- Support and Bug Fixes
- Tips and Documentation



Welcome to **OpenIMPACT** at UIUC,  
an open-source Itanium compiler released under **GELATO**

### News

#### **OpenIMPACT beta release now available ( September 17 2004 )**

The first beta release of OpenIMPACT is [now available!](#)

Gelato members are invited to [contact us](#) for assistance compiling their applications with OpenIMPACT.

#### **Call for participation**

We are expanding the set of programs built and optimized with OpenIMPACT. If your research would benefit from the optimization of any C programs that you run on IA-64 Linux machines, please [contact us](#) and we will help in any way we can. If there are particular parts of any mainstream or specialized library that are stressed particularly hard by your

# OpenIMPACT Release and Support

---

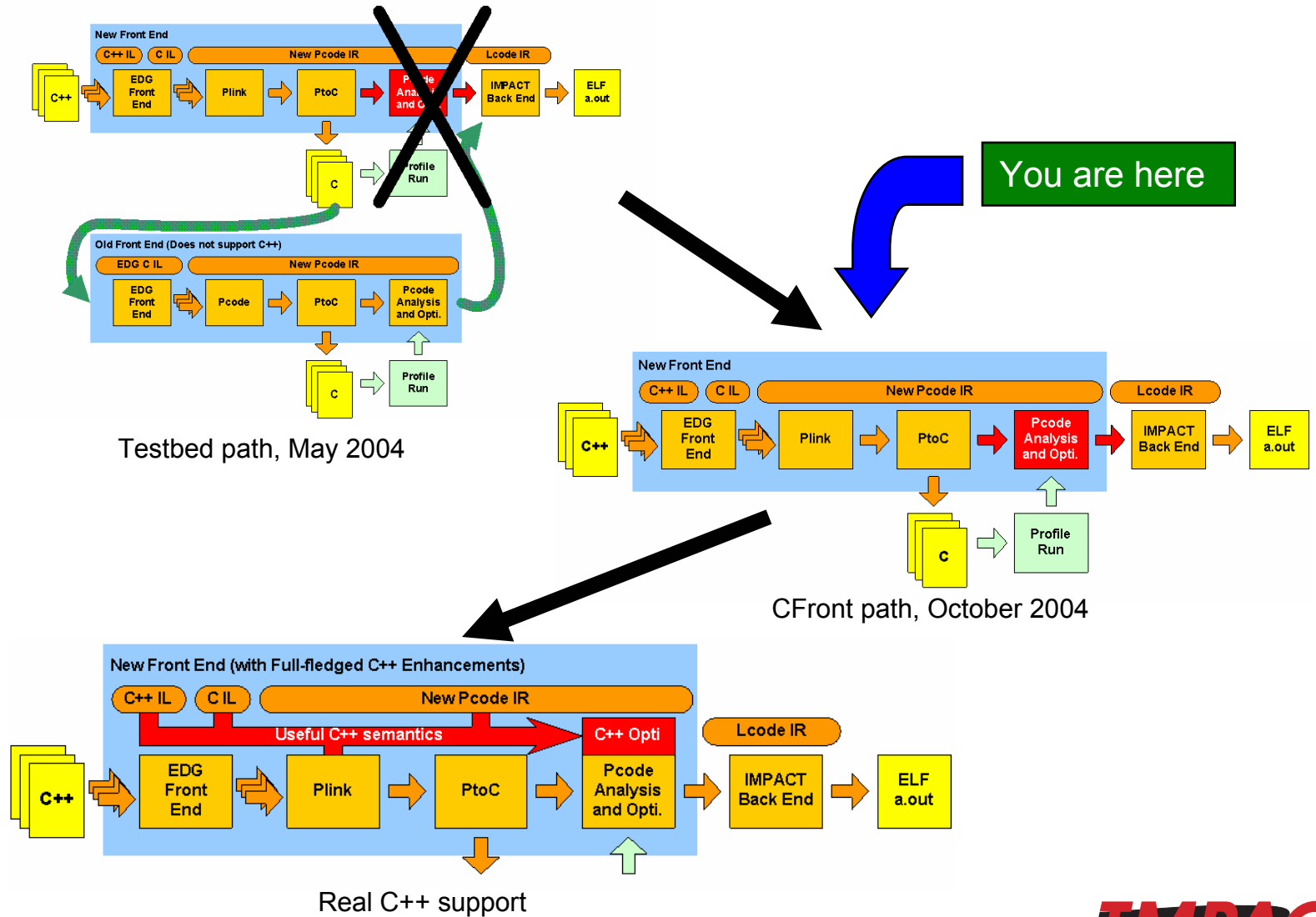
- 40 downloads in first three weeks
  - Testers discovered build system bugs
  - Addressed with 1.0rc1-2004092002 patch
- Thanks to ~~guinea pigs~~ testers at UNSW, DIKU, University of Colorado and Princeton
- [OpenImpact-Support@gelato.uiuc.edu](mailto:OpenImpact-Support@gelato.uiuc.edu)

# C++ Support

---

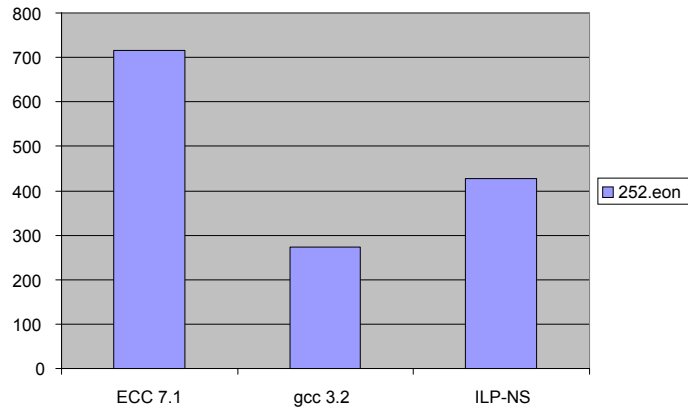
- Front end issues resolved
  - OpenIMPACT can now better handle polymorphism and inheritance
- Driver updated to handle C++
- Baseline compilation of interesting C++ code
  - spec\_namd
  - 252.eon
  - Working on CLHEP

# C++ Progress Map



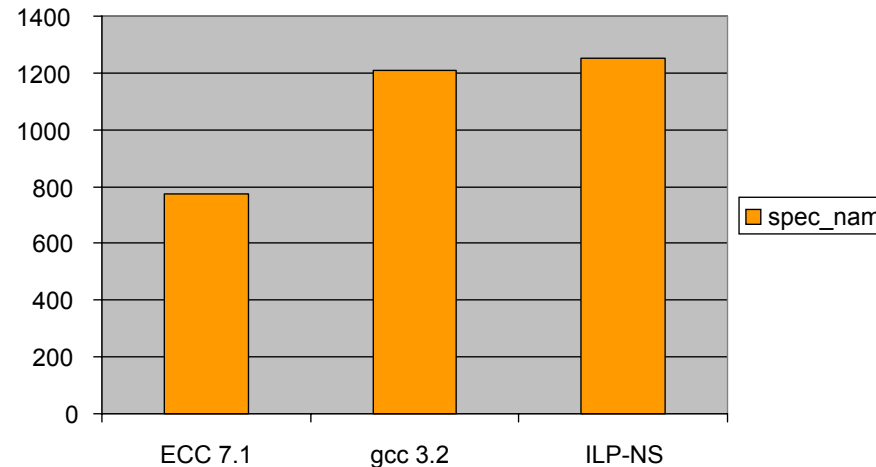
# C++ Performance Comparison

252.eon



Longer is better

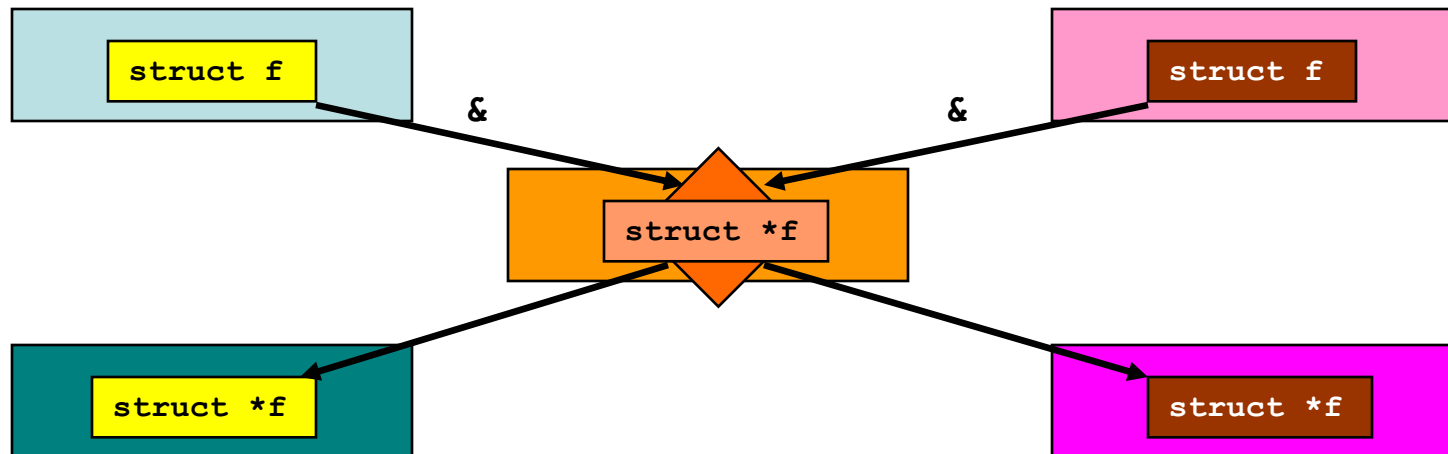
spec\_namd



Shorter is better

# Interesting Challenge - Linker

- OpenIMPACT's linker goes farther than ld
  - Attempts to combine source files into a single, valid abstract syntax tree
- Linker must make sense of types across object files



# Linker

- Problem: Different structs with same name can exist in two source files. Either or both can be used through a struct pointer in a third file.

```
a.c  
  
struct foo {  
    int id;  
    char dummy[8];  
}
```

```
b.c  
  
struct foo {  
    int id;  
    long data;  
}
```

```
dispatcher.c  
  
struct foo *ptr;  
if (test)  
    use_dummy (ptr);  
else  
    use_data (ptr);
```

# Solution



Create unions to hold pointers to each definition of a struct

a.c

```
struct foo_0 {
    int id;
    char dummy[8];
}
```

b.c

```
struct foo_1 {
    int id;
    long data;
}
```

```
union bar {
    struct foo_0 *t_0;
    struct foo_1 *t_1
}
```

```
dispatcher.c

union bar ptr;
if (test)
    use_dummy (ptr);
else
    use_data (ptr);
```

# Solution

- Update functions that care about struct definition to use appropriate union field

```
int use_data (struct foo *f) {  
    ...  
    printf (f->data);  
    ...  
}
```



```
int use_data (union bar f) {  
    ...  
    printf ((f.t_1)->data);  
    ...  
}
```

- Union is passed in the same fashion as a pointer
  - ABI compliant

# Using OpenIMPACT

## oicc provides gcc-like interface

```
% oicc -c foo.c  
% oicc -c bar.c  
% oicc -o program foo.o bar.o
```

## Extracting maximum performance requires some customization

- Kernel patch
- Profiling
- PIP summaries

# Kernel Patch

---

- OpenIMPACT requires a kernel patch to enable general speculation
- Can disable option with `--no-cspec` option
- If kernel patch is not installed, must specify `--no-cspec` option

# Kernel Patch

- Compiling with neither kernel patch nor `--no-cspec` flag can lead to random segfaults in the finished executable
- Debugging will show the segfault occurring at a speculative load instruction

```
{ .mmf // 15
    ld4      loc5 = [loc0]           // op 77 cy 19
=>    ld2.s   in0 = [in0]           // op 86 cy 19 <SM>
    nop.f    0 ;; }                 // op 215 cy 19
```

# Kernel Patch - Future

---

- Investigating method to detect presence of patch
  - Intelligently select default for --no-cspec flag at build time
- Possible integration into Linux tree
  - General Speculation must be made more generally applicable
    - Some code may rely on trapping segfaults; general spec could break this code.

# Profiling

---

- Many optimizations in OpenIMPACT require profile data
- Pcode profiling
  - Generally sufficient
  - --pprof-gen, --pprof-use
- Lcode profiling
  - Some loops are not detected by Pcode profiling
  - Requires a second, slower profiling run
  - --lprof-gen --lprof-use

# Example Profiling Run

- Compile source to object normally

```
oicc -g --verbose -c foo.c
```

- Profiling takes place during linking

```
% oicc -g --verbose --pprof-gen -o program foo.o bar.o  
% ./program.prof input.dat  
% oicc -g --verbose --pprof-use -o program foo.o bar.o
```

- Profiled binary has .prof extension, so Makefile dependencies are typically not satisfied until final binary is written

```
% make CFLAGS=--pprof-gen program  
% ./program.prof input.dat  
% make CFLAGS=--pprof-use program  
% ./program input.dat
```

# Pointer Analysis

- OpenIMPACT's pointer analysis system needs to know how each called function treats its arguments
  - Source code is available for program functions
  - Library functions are black boxes
- Pointer analysis relies on "PIP stubs" to provide details on library functions

# Pointer Analysis



## Missing stubs - symptoms

- Pipa will print warnings for missing stubs

```
% oicc --verbose ... foo.c
...
FUNC DNE [pow]
FUNC DNE [printf]
...
```

- Finished executable may segfault

- Load of memory location may be moved before function call that initializes it

```
foo = alloc ();
init_field (foo);
bar = foo->field;
```



```
foo = alloc ();
bar = foo->field;
init_field (foo);
```

# PIP Stubs

- If library source is available...
  - Compile library with OpenIMPACT
  - OpenIMPACT will extract IR at link time, no need for stubs
  - OpenIMPACT will optimize library code
- But
  - Only works for static (.a) libraries
  - Will increase compile time

# PIP Stubs

- PIP stubs are defined in platform/lib.c
  - Must emulate argument handling
  - Do not need to emulate any other aspect

```
int printf(const char *format, ...)
{
    FILE local;
    int i;
    va_list ap;
    char *p;

    va_start (ap, format);

    *stdout = local;
    for (i = 0; format[i]; i++);

    while ((p = va_arg (ap, char *)))
        *p;

    va_end(ap);
}
```

# Give OpenIMPACT a Try

---

- ⊕ <http://gelato.uiuc.edu>
- ⊕ [OpenImpact-Support@gelato.uiuc.edu](mailto:OpenImpact-Support@gelato.uiuc.edu)
  - Bug reports
  - Compilation assistance