## Integrating Compilers to Support Application Development & Optimization in Eclipse /PTP

University of Houston Oscar Hernandez Barbara Chapman

# The Open64 Compiler

- A robust suite of open source optimizing compiler tools for Linux/Intel IA-64 systems. It's in public domain.
  - Originally developed by SGI, and currently maintained by Hewlett-Packard.
     Other companies have Open64-based products.
  - Full support for F95/F90/F77, C, C++.
  - State-of-the-art analysis and optimizations.
- Our branch of Open64 is called OpenUH:
  - Supports OpenMP 2.0 and optimizations,
  - Improved source-to-source capabilities
  - Tools interfaces, export static analysis, performs instrumentation, runtime library include performance monitoring.
  - Supports automatic parallelization.

## **Integration Efforts**

OpenUH

Pathscale Compiler

OpenMP, Language Extensions, Analysis, Modeling

> Interfaces TAU, **—** KOJAK, Perfsuite,

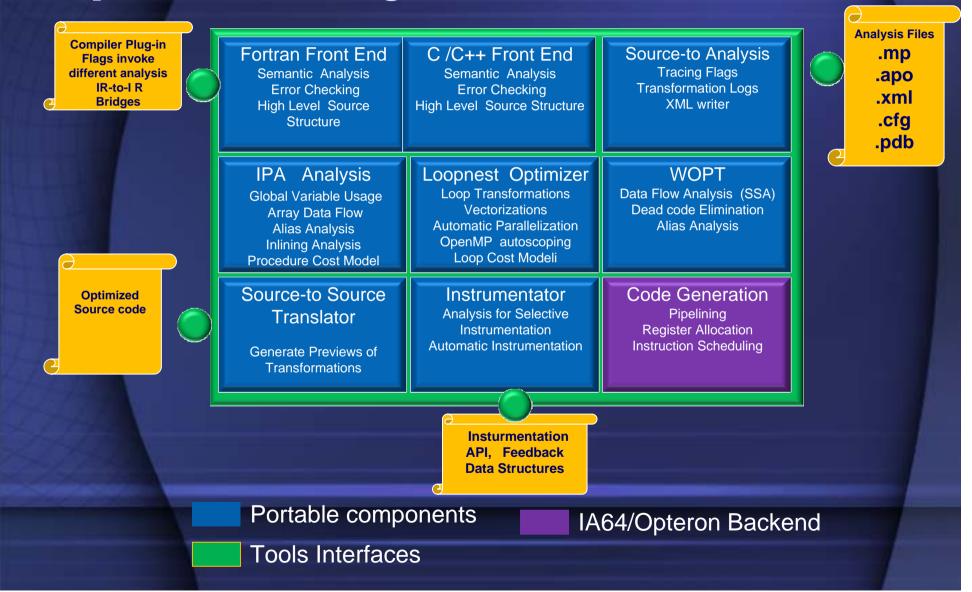
Open64 from Hewlett Packard

Source-to-Source Berkley UPC

 ECLIPSE/PTP plugins

http:www.cs.uh.edu/~openuh

## **Open64 Analysis Infrastructure**



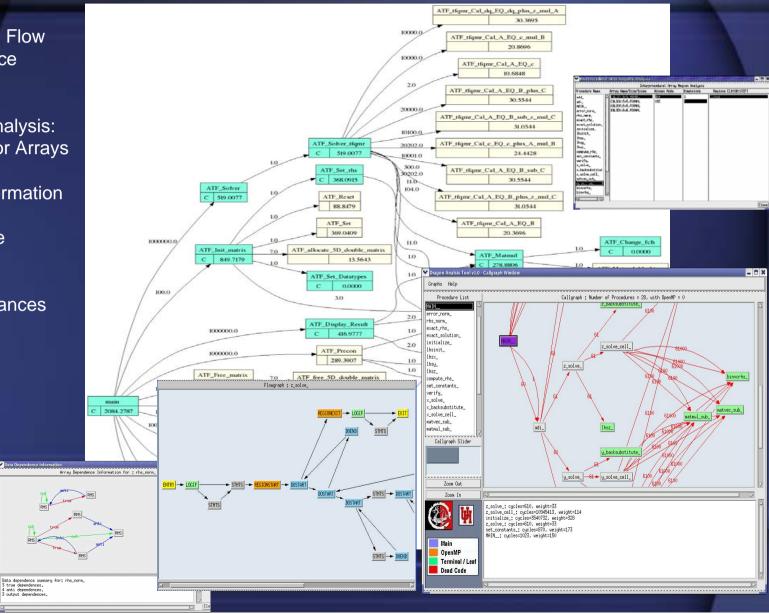
### How compiler analysis can help:

- Be a source of information for the application developer.
- Provide the infrastructure to support development/optimization environments:
  - Semantic Error/Checkers
  - Refactoring Tools
  - Modeling Tools
  - Assist the creation of parallel code (e.g. OpenMP, Hybrid MPI/OpenMP code)
- Support work of other tools.

### Example: Dragon Analysis Tools.

- Callgraph, Control Flow Graph, Dependence Analysis.
- •Interprocedural Analysis: Variable Usage For Arrays
- •Cost Modeling Information
- Detects Expensive Procedures

### •Detect Load Imbalances (MPI/OpenMP)



## **Program Transformations**

**Optimization Logs** 

### Hierarchical Representation of Program

### Procedure MAIN (test) product "Mongoose PARALLELIZATION LOG END Tid=1 version "3.1" begin line=1, end line=12 language "f90" DOACROSS\_LOG\_BEGIN Source Code tid 3 "\_\_mpdo\_MAIN\_1" tid 4 "\_\_mpdo\_MAIN\_2" DOACROSS\_LOG\_END whirl2f "test.m" Analysis Messages Variables list: function 1 "MAIN" range [1 1 0]-[1 1 2 0] yarlist 1 "A"(A):gy, "B"(A):gy, "C"(A):r, "I"(A):gy, "J"(A):gy A(automatic), read, write B(automatic), read, write NEST LOG BEGIN C(automatic), read, write I(automatic), read, write 1 program test can't parallelize: tid 21 J(automatic), read, write oloop 2 "do J" range [130]-[1100] tid 3 2 tid 4 2 Reasons real a(1100),b(1100),c(1100) 2: index "J Array Sections varlist 2 "A"(A):py, "B"(A):py, "C"(A):r, "I"(A):py, "J"(A):py, elogp 3 "do I" range [1 4 0]-[16 0] NEST\_LOG\_END A-> dependence 3: do j=3,1000 POST DOACROSS LOG BEGIN B->B dependence tid 3 "\_\_mpdo\_MAIN\_\_1" tid 4 "\_\_mpdo\_MAIN\_\_2" POST\_DOACROSS\_LOG\_END index "I" varlist 3 "A"(A):w, "B"(A):r, "C"(A):r, "I"(A):rw 4: do i=2. 1100 .1 end\_oloop3 oloop 4 "do I" range [170]-[190] Loop do j=3,1000 5: a(i) = b(i) + c(i)Tid=2 index "I" POST\_NEST\_LOG\_BEGIN 6: enddo begin line=3, end line=10 varlist 4 "A"(A):r, "B"(A):w, "C"(A):r, "I"(A);w tid 2 1 tid 3 2 end\_oloop4 7: do i=2. 1100 .1 Autoparallelization tid 4 2 POST NEST LOG END end\_oloop2 Variables list: end function Transformations 8: b(i) = a(i) + c(i)A(automatic), read, write B(automatic), read, write TRANSFORMATION\_LOG\_BEGIN C(automatic), read, write I(automatic), read, write INTERCHANGE 2,33,2 TRANSFORMATION\_LOG\_END msgloop lines [170] msgloop lines [110] 9: enddo Loop is Parallel J(automatic), read, write Loop msgloop lines [140] 10: enddo Transformation PARALLELIZATION LOG BEGIN 11: write(\*,\*) a(500) tid 2 ARRAY\_DEP highlight "A" lines 1 msgloop lines [140] highlight "A" lines 1 end program v tid 2 ARRAY\_DEP highlight "B" lines 1 12: msgloop lines [170] SRCPOS MAP BEGIN highlight "B" lines 1 tid 2 PARTIAL\_ARRAY\_SEC highlight "A" tid 3 PARALLELIZED Loop do i=2,1100.1 Loop do i=2,1100.1 SRCPOS MAP END Tid=4 Tid=3 tid 4 PARALLELIZED begin line=4, end line=6 begin line=7, end line=9 Variables list: Variables list: A(automatic), read, write B(automatic), read, write A(automatic), read, write B(automatic), read, write Optimized C(automatic), read, write I(automatic), read, write C(automatic), read, write I(automatic), read, write Source Code **Fortran Front** Loop Nest Source-to Inter-Source-to Optimizer / End Procedural Analysis Source WOPT Translator Analysis

# Initial Plug-in for PTP 1.1

### Tool Chain plug-in for OpenUH/Open64.

Configuration Settings	
Tool Settings Build Settings Build	d Steps Error Parsers Binary Parser Environment Macros
Section of the secti	-LNO:opt=1(enabled only if the optimization level of -O3 or higher)      -LNO:build_scalar_reductions=ON(The default setting is OFF)
🖉 Symbols	-LNO:blocking=OFF(The default setting is ON)
2 Directories	Configuration Settings
Configuration Settings Tool Settings Build Settings Build Steps Error Parsers Binary Parser Environment Macros	Tool Settings Build Settings Build Steps Error Parsers Binary Parser Environment Macros
	IPA:aggr_cprop=OFF(The default setting is ON )
Preprocessor All options: -O3 -INLINE -LNO:build_scalar_reductions=	=ON -g - C S Open64 Fortran Compiler 🗍 -IPA:dfe=OFF(The default setting is ON)
2 Symbols	Source □ -IPA:dve=OFF(The default setting is ON)
20 Directories	Symbols
2 Optimization	Directories
Bebugging     Summary and a second seco	Optimization
🖉 Warnings	Optimization(Disabling Options)
Miscellaneous V & Open64 Fortran Compiler Flags Classificat	BLoop Nest Optimization
V \$ Open64 Fortran Compiler Flags Classificat	Optimization(Interprocedural Analysis)
Symbols More than 120	Poptimization(Inline Functions)
<sup>®</sup> Directories	Optimization(Code Generation)
Boptimization Flags Available	@Debugging
Boptimization(Disabling Options)	Warnings
A cop Nest Optimization	2 Miscellaneous
	Defaults Apply
Restore Defaults     Apply	

# Major Challenges

- Portability to support all Eclipse/PTP users.
- Interactions with tools can be complicated (e.g. IR-to-IR mappings, two-way interactions)
- No standard intermediate representation
- No standard format represents output of analyses
- Mappings of analyses to the source code has to be maintained.
- Provide analysis in an intuitive way to the user.
   (e.g. scalability is a problem)

# Conclusions

 We have began to move our compiler to Eclipse/PTP

### • Focus so far:

- Exporting analyses
- Making it easy to invoke the compiler
- Defining interfaces with PTP/other tools

Many standard interfaces seem to be needed.

# **Questions?**

### Example: Conversion to Program Database Representation

