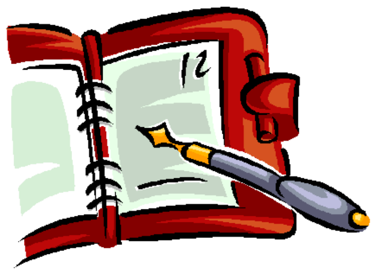
Debugging on the Intel® Xeon Phi™ Coprocessor

Dr.-Ing. Michael Klemm
Software and Services Group
Intel Corporation
(michael.Klemm@intel.com)

Legal Disclaimer & Optimization Notice

Agenda



➔ **Overview**

Installation

Command Line Debugger

- **Debugging a Coprocessor Native Application**
- **Debugging Offloaded Code**
- **The GNU* Project Debugger (GDB*) & Intel® Debugger (IDB)**

Eclipse* CDT Integration

GDB* Enabling

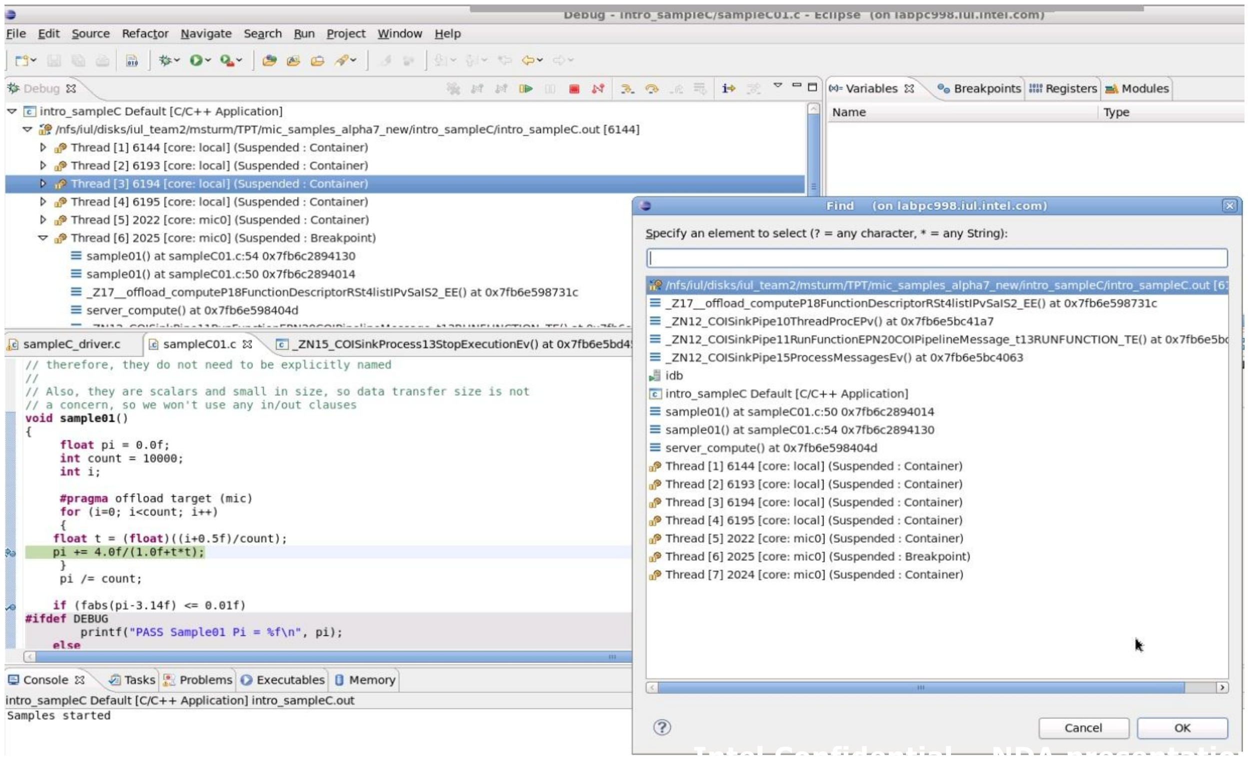
Motivation

Intel® Xeon Phi™ Coprocessor relies on new programming models and debug communication models.

Intel® Debugger provides cross-debug solution to debug on Intel® Xeon Phi™ Coprocessor based coprocessor cards

Intel® Debugger for Intel® Xeon Phi™ Coprocessor

Command Line Debugger
Eclipse* IDE Integration
Linux* hosted
C/C++ & Fortran
Offload and Native Code Debug


Debug - intro_sampleC/sampleC01.c - Eclipse (on labpc998.iul.intel.com)
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Variables
Breakpoints
Registers
Modules
Name
Type
sampleC_driver.c
sampleC01.c
_ZN15_COISinkProcess13StopExecutionEv() at 0x7fb6e5bd4
Console
Tasks
Problems
Executables
Memory
Find (on labpc998.iul.intel.com)
Cancel
OK

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

```c
// therefore, they do not need to be explicitly named
//
// Also, they are scalars and small in size, so data transfer size is not
// a concern, so we won't use any in/out clauses
void sample01()
{
     float pi = 0.0f;
     int count = 10000;
     int i;

     #pragma offload target (mic)
     for (i=0; i<count; i++)
     {
    float t = (float)((i+0.5f)/count);
    pi += 4.0f/(1.0f+t*t);
     }
     pi /= count;

    if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
        printf("PASS Sample01 Pi = %f\n", pi);
    else
```

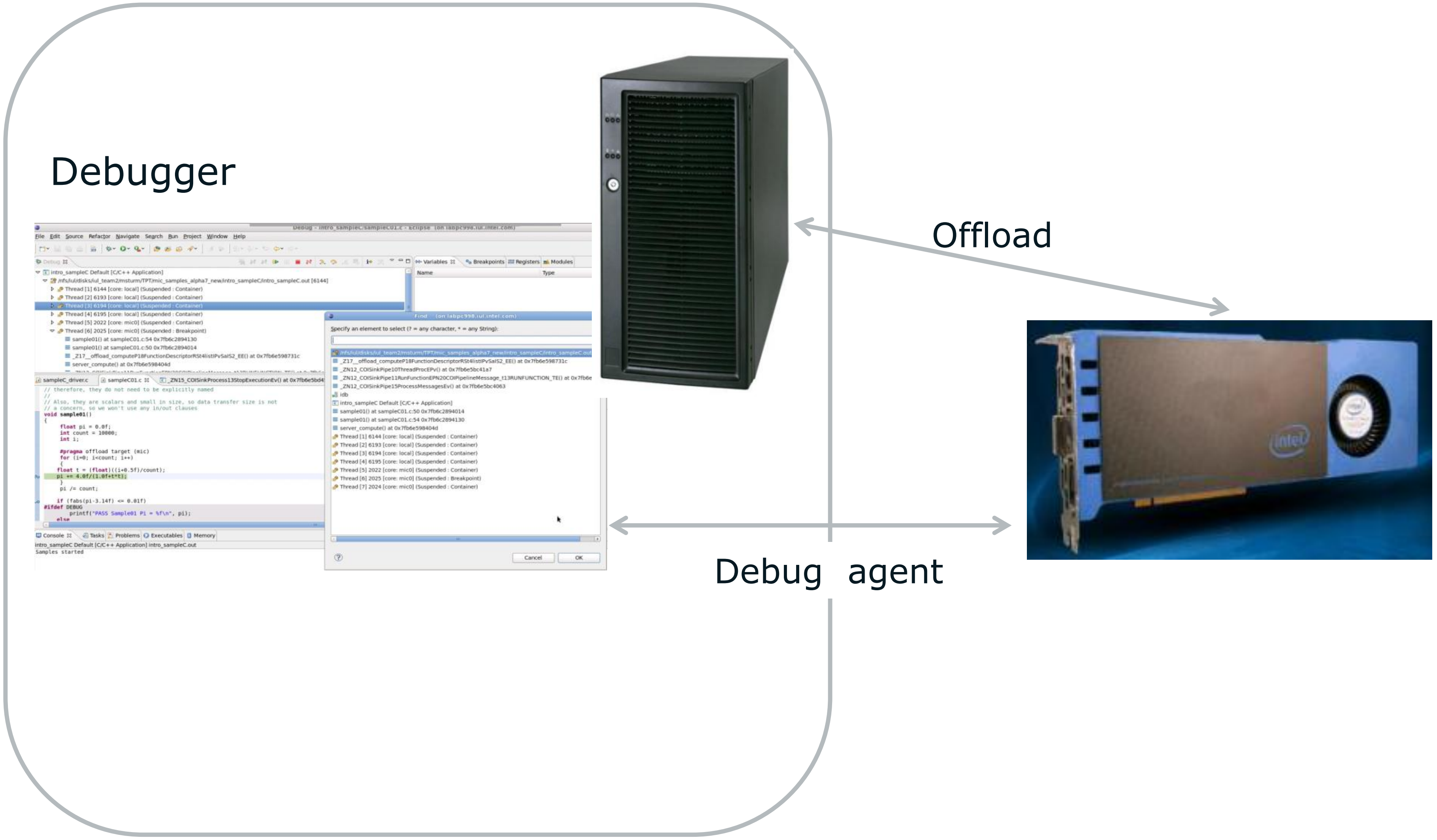
```
intro_sampleC Default [C/C++ Application] intro_sampleC.out
Samples started
```

Specify an element to select (? = any character, * = any String):

```
/nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6
_Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
_ZN12_COISinkPipe10ThreadProcEPv() at 0x7fb6e5bc41a7
_ZN12_COISinkPipe11RunFunctionEPN20COIPipelineMessage_t13RUNFUNCTION_TE() at 0x7fb6e5bc
_ZN12_COISinkPipe15ProcessMessagesEv() at 0x7fb6e5bc4063
idb
intro_sampleC Default [C/C++ Application]
sample01() at sampleC01.c:50 0x7fb6c2894014
sample01() at sampleC01.c:54 0x7fb6c2894130
server_compute() at 0x7fb6e598404d
Thread [1] 6144 [core: local] (Suspended : Container)
Thread [2] 6193 [core: local] (Suspended : Container)
Thread [3] 6194 [core: local] (Suspended : Container)
Thread [4] 6195 [core: local] (Suspended : Container)
Thread [5] 2022 [core: mic0] (Suspended : Container)
Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
Thread [7] 2024 [core: mic0] (Suspended : Container)
```

Optimization Notice

Debugger for Intel® Xeon Phi™ Coprocessor Basic Setup


Debugger
Offload
Debug agent
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Variables Breakpoints Registers Modules
Name Type
Console Tasks Problems Executables Memory
Cancel OK

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

sampleC_driver.c | sampleC01.c | _ZN15_COISinkProcess13StopExecutionEv() at 0x7fb6e5bd4

```c
// therefore, they do not need to be explicitly named
//
// Also, they are scalars and small in size, so data transfer size is not
// a concern, so we won't use any in/out clauses
void sample01()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    for (i=0; i<count; i++)
    {
   float t = (float)((i+0.5f)/count);
   pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;

   if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
       printf("PASS Sample01 Pi = %f\n", pi);
   else
```

```
intro_sampleC Default [C/C++ Application] intro_sampleC.out
Samples started
```

Find (on labpc990.iul.intel.com)

Specify an element to select (? = any character, * = any String):

```
/nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out
_Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
_ZN12_COISinkPipe10ThreadProcEPv() at 0x7fb6e5bc41a7
_ZN12_COISinkPipe11RunFunctionEPN20COIPipelineMessage_t13RUNFUNCTION_TE() at 0x7fb6e
_ZN12_COISinkPipe15ProcessMessagesEv() at 0x7fb6e5bc4063
idb
intro_sampleC Default [C/C++ Application]
sample01() at sampleC01.c:50 0x7fb6c2894014
sample01() at sampleC01.c:54 0x7fb6c2894130
server_compute() at 0x7fb6e598404d
Thread [1] 6144 [core: local] (Suspended : Container)
Thread [2] 6193 [core: local] (Suspended : Container)
Thread [3] 6194 [core: local] (Suspended : Container)
Thread [4] 6195 [core: local] (Suspended : Container)
Thread [5] 2022 [core: mic0] (Suspended : Container)
Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
Thread [7] 2024 [core: mic0] (Suspended : Container)
```

Optimization Notice

Debugger Installation

Intel® Debugger for Intel® Xeon Phi™ Coprocessor is part of the Intel® Composer XE for Linux* Including Intel® Xeon Phi™ Coprocessor.

Gets installed automatically running the install.sh install script from the l_ccompxe_2013.0.xxx.tgz package.

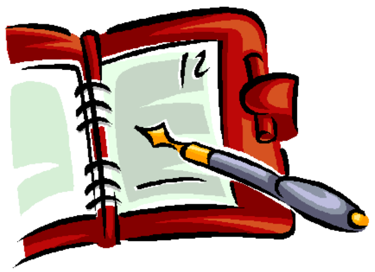
Setting up the Intel® C++ Compiler environment via

```
$ source /opt/intel/composer_xe_2013/bin/compilervars.sh
intel64
```

will also set up the environment for the Intel® Debugger.

Automatically installed as part of Intel® Composer XE
Integration into Eclipse* CDT covered later

Agenda



Overview

Installation

➔ **Command Line Debugger**

- **Debugging a Coprocessor Native Application**
- Debugging Offloaded Code
- The GNU* Project Debugger (GDB*) & Intel® Debugger (IDB)

Eclipse* CDT Integration

GDB* Enabling

Optimization Notice

Command Line Debugging

Debugger Executables

After installation is complete you will find the debugger executables at

`/opt/intel/composer_xe_2013/bin/intel64_mic`

`idbc` is the command line debugger driver for the host.

`idbc_mic` is the command line debugger driver for the Intel® Xeon Phi™ Coprocessor based card.

idbc and idbc_mic are the debugger executables

Launching the Debugger

Launch the target debugger by using the following command:

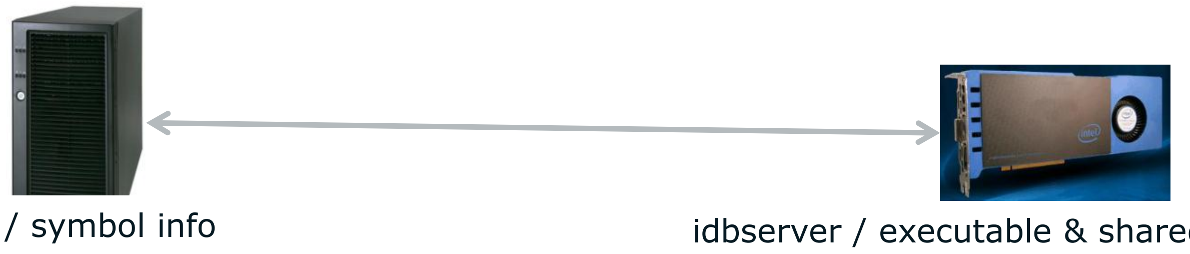
```
idbc_mic -tco -rconnect=tcpip:coprocessor-ip-address:port
```

or

```
idbc_mic -tco -rconnect=tcpip:mic[n]:port
```

when using the default IP addresses.

The default port number is 2000.



For example, enter

```
idbc_mic -tco -rconnect=tcpip:mic0:2000
```

for the first Intel® Xeon Phi™ Coprocessor device using the default port.

Starting Application from within Debugger

If you are debugging an application to run natively on Intel® Xeon Phi™ Coprocessor you can start the application using the debugger:

1. Specify the remote executable file:

(idb) idb file-remote *target-bin-path*

2. Specify the file containing debug info on host:

(idb) file *host-bin-path*

3. After setting breakpoints and whatever else you want to do before starting the application, start the
Application to reach breakpoint:

```
(idb)run
```

Note:
The application has to be uploaded to the coprocessor outside of the debugger prior to launching it.

Attaching to Application with Debugger

Attach

If you are debugging an application on the coprocessor target, run the application and attach the debugger to it:

Enter the following command:

```
(idb) attach <pid> <image_file>
```

<pid>	The pid of the process to attach to.
<image_file>	The image path and file name on the host

Target Process List

To identify the process to attach to the following command can be used

```
(idb) idb show process-list ["proc-name"]
```

Semantics:
`idb show process-list` prints all processes running on the target and variable `$lsproc` will hold the number of processes found as a negative number, i.e. If there are 5 processes running, `$lsproc` will be -5

`idb show process-list "proc-name"` will get the process list and search for a process named proc-name. If found, `$lsproc` will contain the process id or 0 if no applicable process has been found.

Native Debug Considerations (1)

1. New command:

Setting remote working directory on the coprocessor:

```
(idb) idb remote-working-dir
```

2. MPSS supports standard Linux user account configurations

idbserver_mic and the native application to be debugged need to be launched as the same user.

This means both the native app on the target and idc_mic need to either be launched as root or with a user account of the same name.

Native Debug Considerations (2)

When debugging native coprocessor applications on the command line, the shared library libmyodbl-service.so, needs to be uploaded manually moving forward.

Solution:

Create an overlay, so the file is uploaded at boot time. Follow the instructions on how to use overlays in the MPSS readme.txt.

Specific Steps:

a. Target: Create /etc/sysconfig/mic/conf.d/myo.conf containing

```
# MYO download files
Overlay / /opt/intel/mic/myo/config/myo.filelist
```

b. Host: Create /opt/intel/mic/myo/config/myo.filelist containing

```
dir /lib64 755 0 0
file /lib64/libmyodbl-service.so
opt/intel/mic/myo/lib/libmyodbl-service.so 755 0 0
```

Using an Endless Worker Loop (1)

It may be useful to use an endless worker loop to be able to attach to an application at a defined location.

To ensure the endless worker loop in your native or coprocessor-side application is executed even with aggressive compiler optimizations enabled it is recommended to implement it as follows:

```
void attach_idb() {
    volatile int loop = 1;
    do {
        volatile int a = 1;
    } while (loop);
}
```

Call this function at a location suitable for attaching the debugger.

Using an Endless Worker Loop (2)

If you added an endless worker loop to your offload code, the following method may be used to start debugging just outside of the loop.

1. Using the following commands:

```
(idb) list <filename>
```

2. Now you have a line number for `volatile int a =1.` Set a breakpoint on that line and continue.

```
(idb) p loop=0
(idb) break <line number>
```

3. Continue

```
(idb) continue
```

You can identify the source line directly after the endless worker loop and start debugging from that source line.

Note: After that, set a breakpoint using the break command at any code line or address of interest and issue a continue command to run to it.

Shared Object Debugging

Using LD_LIBRARY_PATH

If the target-side application is dynamically linked against shared objects that are not part of the compiler or driver, and that need `LD_LIBRARY_PATH` to be set in order to find them,

1. Set a breakpoint before the libraries are loaded.
2. When the debugger stops at that location, use the command

```
(idb) set environment LD_LIBRARY_PATH path
```

3. Do the same for other environment variables that the application might need.

Specifying Additional Symbol Info Search Paths

To tell the debugger where to search for the debug information specific to your application:

```
(idb) set solib-search-path path[:path]
```

e.g.

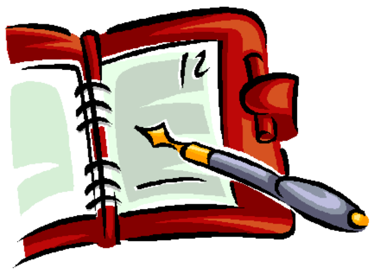
```
(idb) set solib-search-path /usr/linux-k1om-4.7/linux-k1om/lib64/lib:/usr/linux-
k1om-4.7/x86_64-k1om-linux/lib64
```

To ensure correct target Linux* runtime library pick-up by the debugger

```
(idb) show solib-search-path
```

Provides listing of all directories in search path

Agenda



Overview

Installation

Command Line Debugger

- Debugging a Coprocessor Native Application
- ➔ **Debugging Offloaded Code**
- The GNU* Project Debugger (GDB*) & Intel® Debugger (IDB)

Eclipse* CDT Integration

GDB* Enabling

Simultaneous Debug Host and Coprocessor (1)

No Debug Synchronization between Host and Coprocessor

For command line debug there is no active synchronization between host and coprocessor debugged code.

1. Set your host side breakpoint where a target workload already exists.
2. After attach with idbc_mic, set a breakpoint there where you want to start debugging.

If you are debugging a heterogeneous application and intend to debug host and coprocessor code simultaneously,

run idbc for the host

run idbc_mic for the coprocessor targeted codebase.

➔ Two terminal windows or for a remote debug setup two ssh sessions will be necessary.

Simultaneous Debug Host and Coprocessor (2)

Host Debug:

```
idbc <application>
```

Start the app on the host side through the debugger and stopped at a breakpoint somewhere after creating target offload process.

Target Debug:

```
idbc_mic -tco -rconnect=tcpip:<cardip>:<port>
idbc_mic -tco -rconnect=tcpip:mic[n]:<port>

(idb) attach <pid> /opt/intel/composerxe/lib/mic/offload_main
```

The actual location of the `offload_main` binary may differ depending on the tools version used.

Simultaneous Debug Host and Coprocessor (3)

Target Debug:

IDB will attach to the process and read debug info from debuggee process and loaded libraries. If the libraries are located at a different location than at compile time, you can set up library search paths using the debugger command.

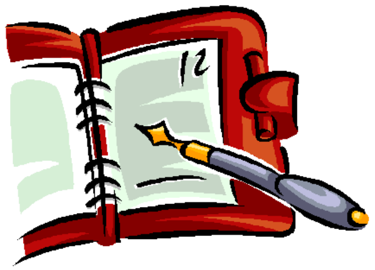
```
(idb) set solib-search-path <path-to-so>[:<path-to-so>]
```

Note:

The location the debugger stops on the target is random, typically in the scheduler or libpthread:
- set host side breakpoint where target workload already exists
- after attach with idbc_mic set a breakpoint there where you want to start debugging.

- **You may want to consider introducing infinite worker loop in offload code do define connection point.**

Agenda



Overview

Installation

Command Line Debugger

- Debugging a Coprocessor Native Application
- Debugging Offloaded Code
- ➔ The GNU* Project Debugger (GDB*) & Intel® Debugger (IDB)

Eclipse* CDT Integration

GDB* Enabling

Intel® Debugger (IDB) and The GNU* Project Debugger (GDB*)

IDB is the Intel debugger. It has a GDB-style command line interface

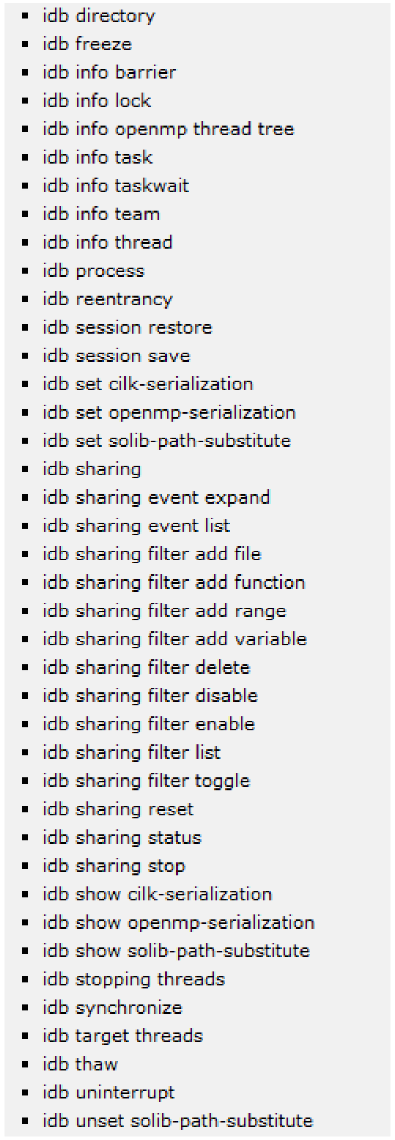
Basic commands and behavior are the same as GDB.

In addition it features,

- Enhanced Fortran 90/95 support
- Support for dynamic arrays in Fortran
- Integration into Eclipse* CDT offers enhanced threading support
- Enhanced Parallelism and Threading Support (next slide)

Debugger Online Help:
http://software.intel.com/sites/products/documentation/hpc/composerxe/en-us/2011Update/idbxe/linux/index.htm

Intel® Debugger unique commands



- idb directory
- idb freeze
- idb info barrier
- idb info lock
- idb info openmp thread tree
- idb info task
- idb info taskwait
- idb info team
- idb info thread
- idb process
- idb reentrancy
- idb session restore
- idb session save
- idb set cilk-serialization
- idb set openmp-serialization
- idb set solib-path-substitute
- idb sharing
- idb sharing event expand
- idb sharing event list
- idb sharing filter add file
- idb sharing filter add function
- idb sharing filter add range
- idb sharing filter add variable
- idb sharing filter delete
- idb sharing filter disable
- idb sharing filter enable
- idb sharing filter list
- idb sharing filter toggle
- idb sharing reset
- idb sharing status
- idb sharing stop
- idb show cilk-serialization
- idb show openmp-serialization
- idb show solib-path-substitute
- idb stopping threads
- idb synchronize
- idb target threads
- idb thaw
- idb uninterrupt
- idb unset solib-path-substitute

Start with "idb"

Cover thread specific run-control

- Define thread groups, freeze, thaw

Intel® Cilk™ Plus and OpenMP execution serialization

Data sharing event detection

Thread filtering

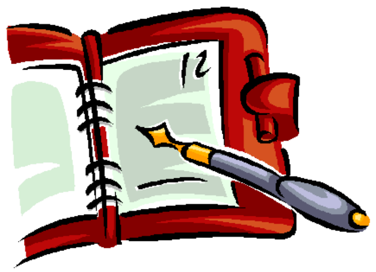
OpenMP* thread info:
- Locks, barriers, teams, tasks, thread tree

idb show process-list "<image-name>"
- Displays process ID of image name.

IDB provides advanced thread run-control and awareness

Optimization Notice

Agenda



Overview

Installation

Command Line Debugger

- Debugging a Coprocessor Native Application
- Debugging Offloaded Code
- The GNU* Project Debugger (GDB*) & Intel® Debugger (IDB)

➔ **Eclipse* CDT Integration**

GDB* Enabling

Optimization Notice

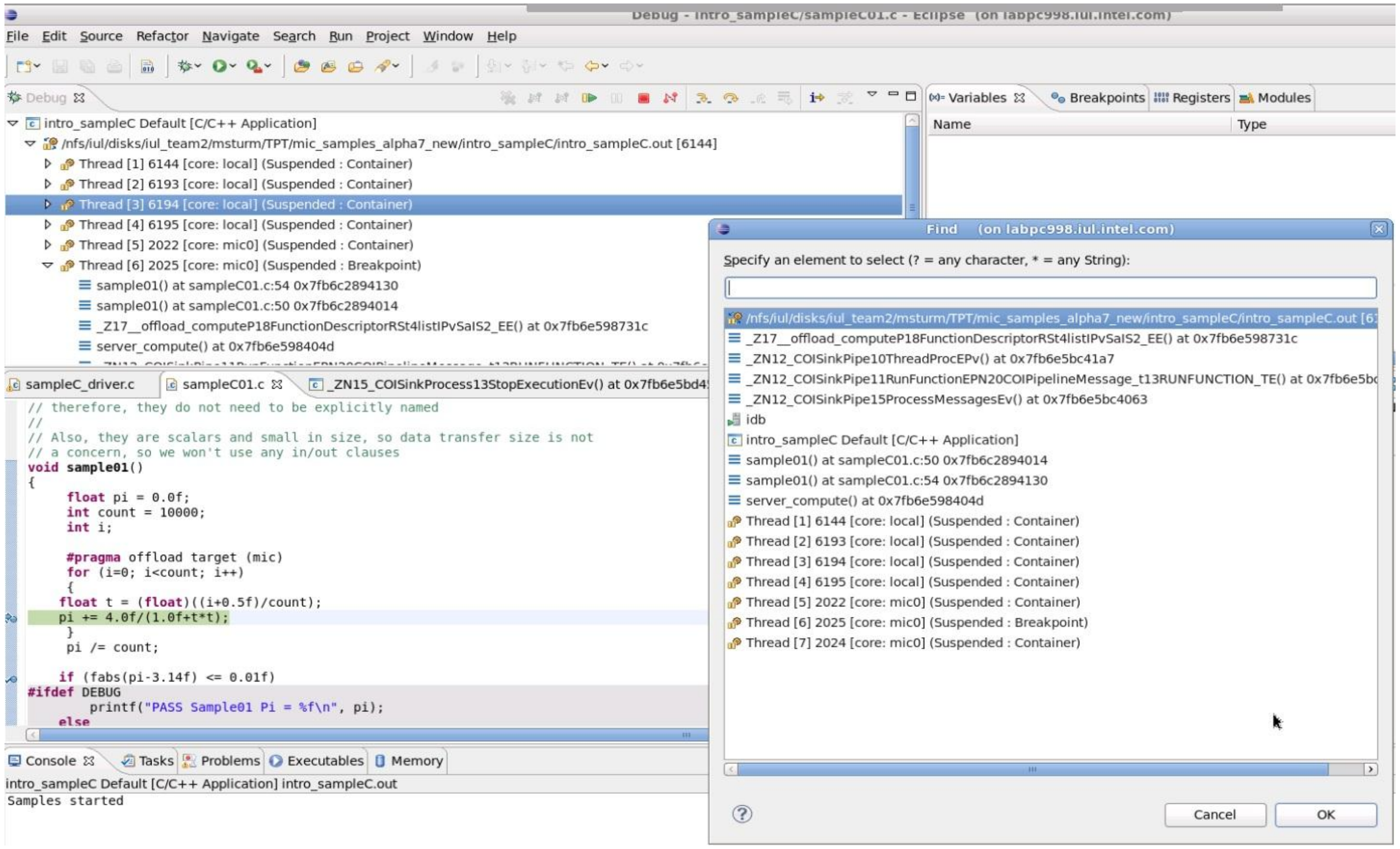
Debugging Heterogeneous Applications in Eclipse* IDE integration

- Offload process gets launched on Intel® Xeon Phi™ Coprocessor. This process executes offloaded code segment.
- Debugger remotely attaches to offload process offload_main
  - Automatic debug agent download on offload
  - Protocol: Virtual TCP/IP
  - Automatic attach to offload process
- Once launching host process is finished, offload process is automatically removed on coprocessor.
- Debugger captures process removal and detaches.

```
main()
{

#pragma offload target (mic)

#pragma omp parallel for reduction(+:pi)
   for (i=0; i<count; i++)
      {
         float t = (float)((i+0.5)/count);
         pi += 4.0/(1.0+t*t);
      }
   pi /= count

}
```

Optimization Notice

Debugging Intel® Xeon Phi™ Coprocessor applications with Eclipse* CDT


Debug - intro_sampleC/sampleC01.c - Eclipse (on labpc998.iul.intel.com)
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Variables
Breakpoints
Registers
Modules
Name
Type
sampleC_driver.c
sampleC01.c
_ZN15_COISinkProcess13StopExecutionEv() at 0x7fb6e5bd4
Console
Tasks
Problems
Executables
Memory
Find (on labpc998.iul.intel.com)
Cancel
OK

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/mstum/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

```
// therefore, they do not need to be explicitly named
//
// Also, they are scalars and small in size, so data transfer size is not
// a concern, so we won't use any in/out clauses
void sample01()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    for (i=0; i<count; i++)
    {
   float t = (float)((i+0.5f)/count);
   pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;

   if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
      printf("PASS Sample01 Pi = %f\n", pi);
   else
```

```
intro_sampleC Default [C/C++ Application] intro_sampleC.out
Samples started
```

Specify an element to select (? = any character, * = any String):

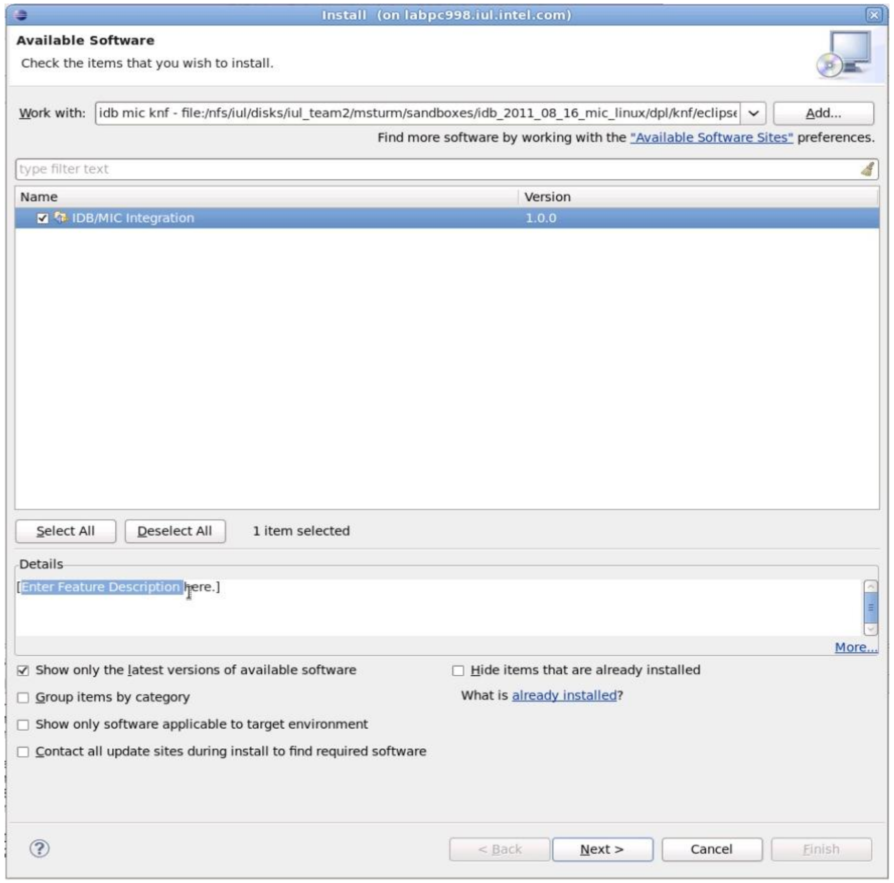
```
/nfs/iul/disks/iul_team2/mstum/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6
_Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
_ZN12_COISinkPipe10ThreadProcEPv() at 0x7fb6e5bc41a7
_ZN12_COISinkPipe11RunFunctionEPN20COIPipelineMessage_t13RUNFUNCTION_TE() at 0x7fb6e5bc
_ZN12_COISinkPipe15ProcessMessagesEv() at 0x7fb6e5bc4063
idb
intro_sampleC Default [C/C++ Application]
sample01() at sampleC01.c:50 0x7fb6c2894014
sample01() at sampleC01.c:54 0x7fb6c2894130
server_compute() at 0x7fb6e598404d
Thread [1] 6144 [core: local] (Suspended : Container)
Thread [2] 6193 [core: local] (Suspended : Container)
Thread [3] 6194 [core: local] (Suspended : Container)
Thread [4] 6195 [core: local] (Suspended : Container)
Thread [5] 2022 [core: mic0] (Suspended : Container)
Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
Thread [7] 2024 [core: mic0] (Suspended : Container)
```

- Eclipse* IDE integrated debugger with integrated thread view and source view for coprocessor code execution.
- Multi-card offload debug. Single card offload and direct/native mode support.

Single Eclipse* GUI based Debug Solution

Standard Eclipse IDE Debugger with integrated Cross-Debug for heterogeneous applications

Install Integration Add-on for Eclipse* >



Install (on labpc998.iul.intel.com)

Available Software

Check the items that you wish to install.

Work with: idb mic knf - file:/nfs/iul/disks/iul_team2/msturm/sandboxes/idb_2011_08_16_mic_linux/dpl/knf/eclipse

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ IDB/MIC Integration	1.0.0

1 item selected

Details

[Enter Feature Description here.]

- ☑ Show only the latest versions of available software
- ☐ Group items by category
- ☐ Show only software applicable to target environment
- ☐ Contact all update sites during install to find required software
- ☐ Hide items that are already installed

What is already installed?

Optimization Notice

Adding the Compiler and Debugger to Eclipse*

1. Start Eclipse.

2. Select or create a workspace. For example, select or create a makefile project with already existing code.

3. Select **Help > Install New Software.**

4. Next to the Work with field, Click the **Add** button.
The **Add Site** dialog opens.

5. Click the **Local** button and browse to the appropriate Intel CDT integration directory:

install_dir/eclipse_support/cdt8.0/eclipse

6. Click **OK**.

7. Make sure **Group items by category** is not checked.

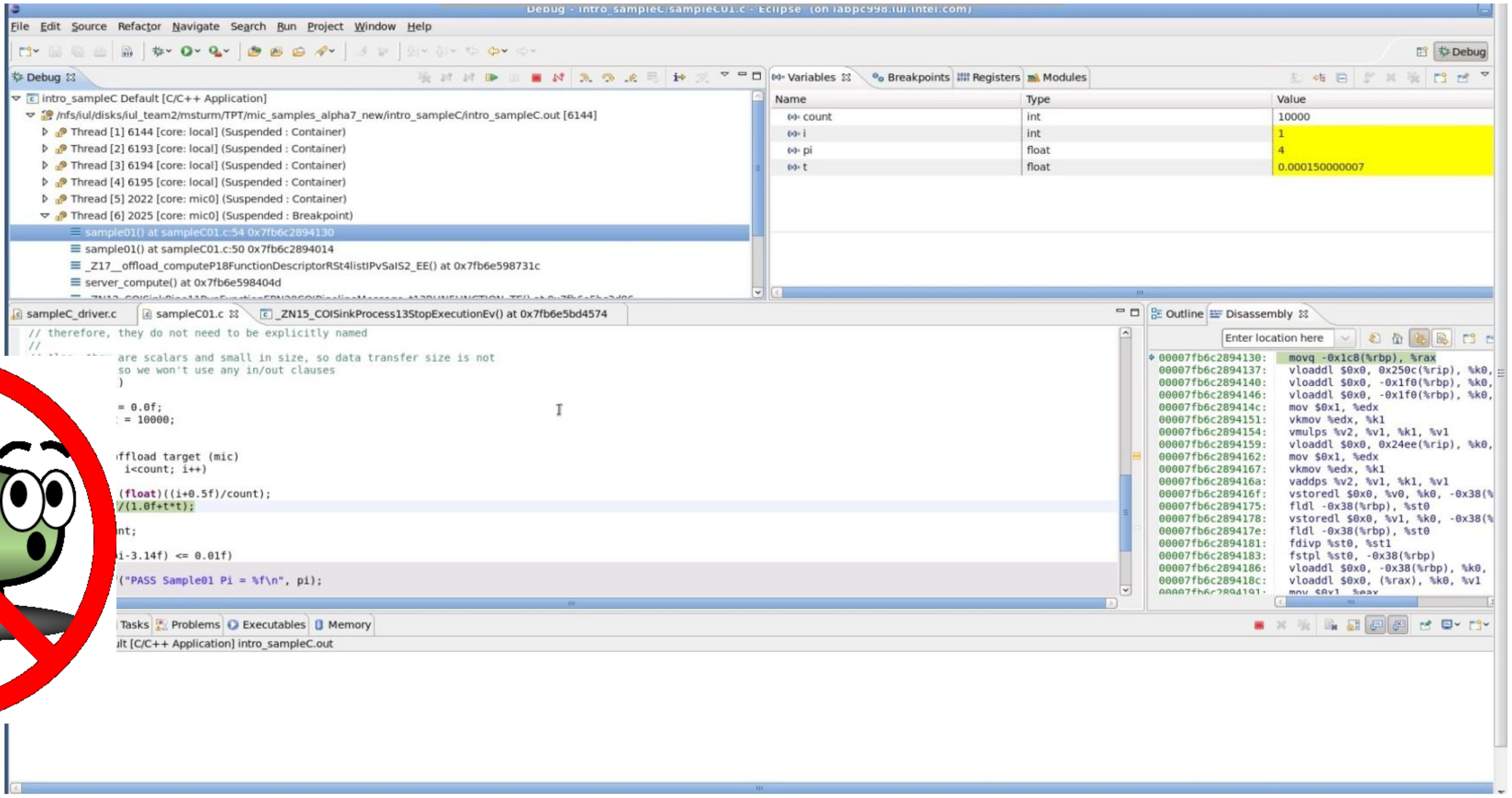
8. Select the options beginning with Intel, including the Intel Debugger (IDB), should you choose to use it, and
click **Next**.

Optimization Notice

Start Offload Debug Session in Eclipse* IDE

Launch Eclipse*: `$ ./eclipse &`

```
labpc998 ~> cd TPT/mic_samples_alpha7_new/
celo_sampleC/        celo_sampleCPP/      hs_err_pid26656.log  intro_sampleC/       .metadata/
labpc998 ~> cd TPT/mic_samples_alpha7_new/
celo_sampleC/        celo_sampleCPP/      hs_err_pid26656.log  intro_sampleC/       .metadata/
labpc998 ~> cd TPT/mic_samples_alpha7_new/
labpc998 ~/TPT/mic_samples_alpha7_new> /users/msturm/EclipseIndigo/MICCDT/eclipse/eclipse &
```


Debug - intro_sampleC/sampleC01.c - Eclipse (on labpc998.iul.intel.com)
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Variables
Breakpoints
Registers
Modules
Outline
Disassembly
Enter location here
Tasks
Problems
Executables
Memory

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

Name	Type	Value
count	int	10000
i	int	1
pi	float	4
t	float	0.000150000007

sampleC_driver.c | sampleC01.c | _ZN15_COISinkProcess13StopExecutionEv() at 0x7fb6e5bd4574

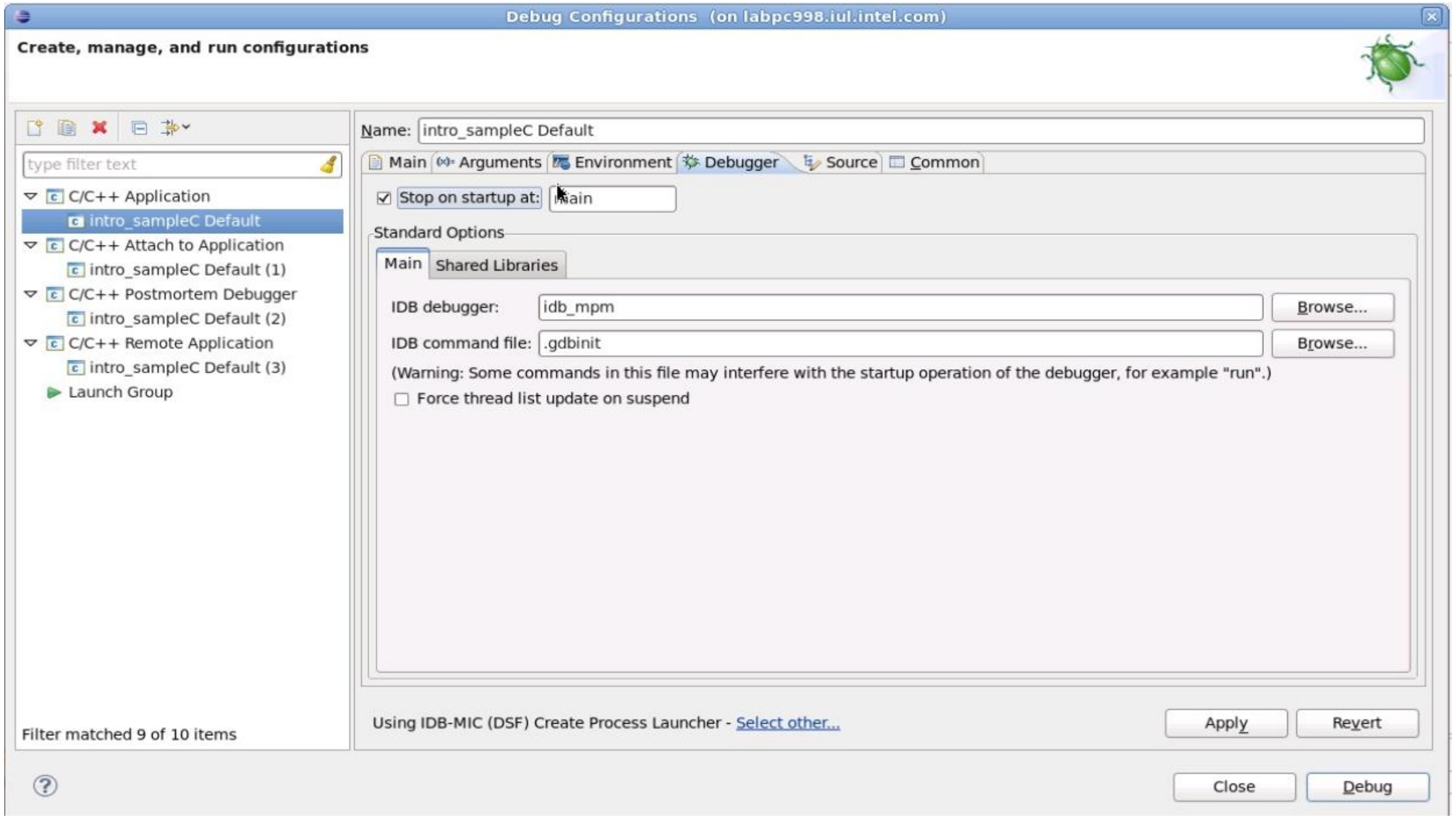
```
// therefore, they do not need to be explicitly named
//
          are scalars and small in size, so data transfer size is not
          so we won't use any in/out clauses
          )

          = 0.0f;
          = 10000;


          ffload target (mic)
           i<count; i++)

          (float)((i+0.5f)/count);
          /(1.0f+t*t);

          nt;

          i-3.14f) <= 0.01f)

          ("PASS Sample01 Pi = %f\n", pi);
```

```
00007fb6c2894130:   movq -0x1c8(%rbp), %rax
00007fb6c2894137:   vloaddl $0x0, 0x250c(%rip), %k0,
00007fb6c2894140:   vloaddl $0x0, -0x1f0(%rbp), %k0,
00007fb6c2894146:   vloaddl $0x0, -0x1f0(%rbp), %k0,
00007fb6c289414c:   mov $0x1, %edx
00007fb6c2894151:   vkmov %edx, %k1
00007fb6c2894154:   vmulps %v2, %v1, %k1, %v1
00007fb6c2894159:   vloaddl $0x0, 0x24ee(%rip), %k0,
00007fb6c2894162:   mov $0x1, %edx
00007fb6c2894167:   vkmov %edx, %k1
00007fb6c289416a:   vaddps %v2, %v1, %k1, %v1
00007fb6c289416f:   vstoredl $0x0, %v0, %k0, -0x38(%
00007fb6c2894175:   fldl -0x38(%rbp), %st0
00007fb6c2894178:   vstoredl $0x0, %v1, %k0, -0x38(%
00007fb6c289417e:   fldl -0x38(%rbp), %st0
00007fb6c2894181:   fdivp %st0, %st1
00007fb6c2894183:   fstpl %st0, -0x38(%rbp)
00007fb6c2894186:   vloaddl $0x0, -0x38(%rbp), %k0,
00007fb6c289418c:   vloaddl $0x0, (%rax), %k0, %v1
```

```
ult [C/C++ Application] intro_sampleC.out
```

Define Debug Configuration – Eclipse* Style

1. Select **Run > Debug Configurations….**
2. Select the debug configuration type, for example, **C/C++ Application.**
3. Click the **New button.**
4. Enter a name for your configuration.



Debug Configurations (on labpc998.iul.intel.com)
Create, manage, and run configurations
type filter text
C/C++ Application
intro_sampleC Default
C/C++ Attach to Application
intro_sampleC Default (1)
C/C++ Postmortem Debugger
intro_sampleC Default (2)
C/C++ Remote Application
intro_sampleC Default (3)
Launch Group
Filter matched 9 of 10 items
Main
Arguments
Environment
Debugger
Source
Common
Standard Options
Main
Shared Libraries
Browse...
Browse...
Select other...
Apply
Revert
Close
Debug

Name: intro_sampleC Default

☑ Stop on startup at: main

IDB debugger: idb_mpm

IDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

☐ Force thread list update on suspend

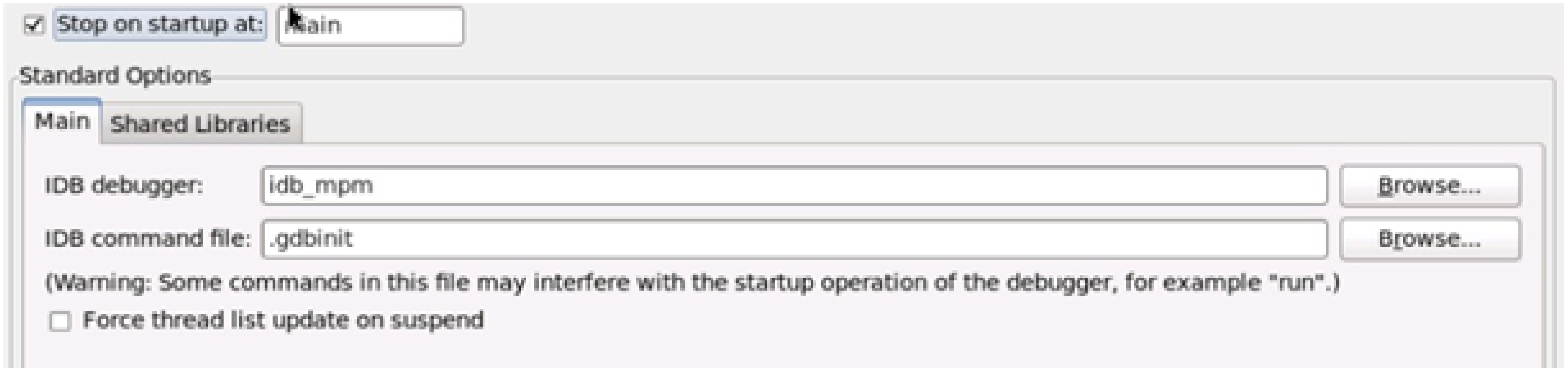
Using IDB-MIC (DSF) Create Process Launcher - Select other...

Define Debug Configuration – Eclipse* Style

1. Switch to the **Debugger tab.**

2. Under **Debugger Options** select the **Main tab**. Make sure that the path to the debugger executable is specified correctly in the field **IDB Debugger.**

3. By default, **IDB debugger contains the correct start script `idb_mpm`.**



☑ Stop on startup at: Main

Standard Options

IDB debugger: idb_mpm

IDB command file: .gdbinit

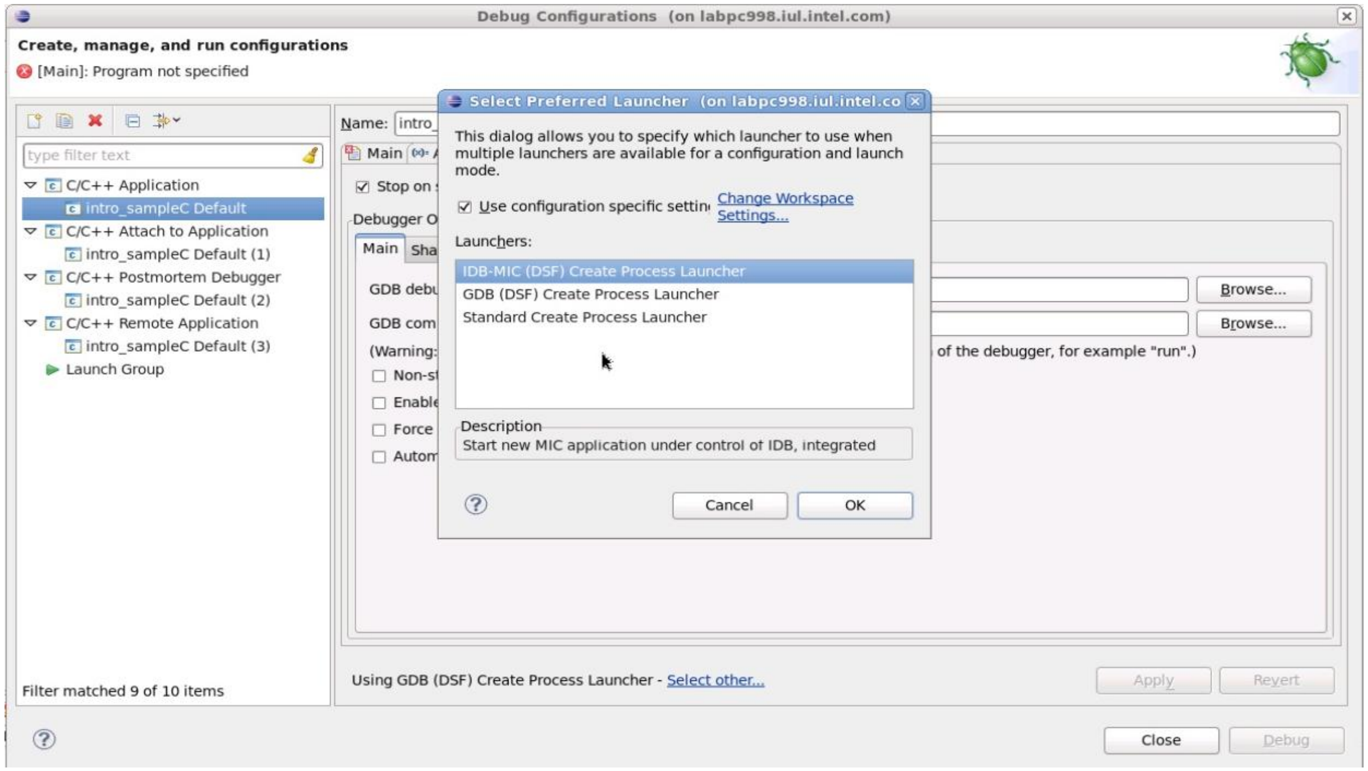
(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

☐ Force thread list update on suspend

Optimization Notice

Select Offload Process Launcher for Debug Session

1. On the **Main tab** locate the process launcher information and click Select other....
2. The **Select Preferred Launcher** dialog box appears.
3. Check the **Use configuration specific settings** checkbox.
4. Select **IDB-MIC (DSF) Create Process Launcher.**



Debug Configurations (on labpc998.iul.intel.com)

Create, manage, and run configurations

[Main]: Program not specified

Select Preferred Launcher (on labpc998.iul.intel.co

This dialog allows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.

☑ Use configuration specific settin Change Workspace Settings...

Launchers:

- IDB-MIC (DSF) Create Process Launcher
- GDB (DSF) Create Process Launcher
- Standard Create Process Launcher

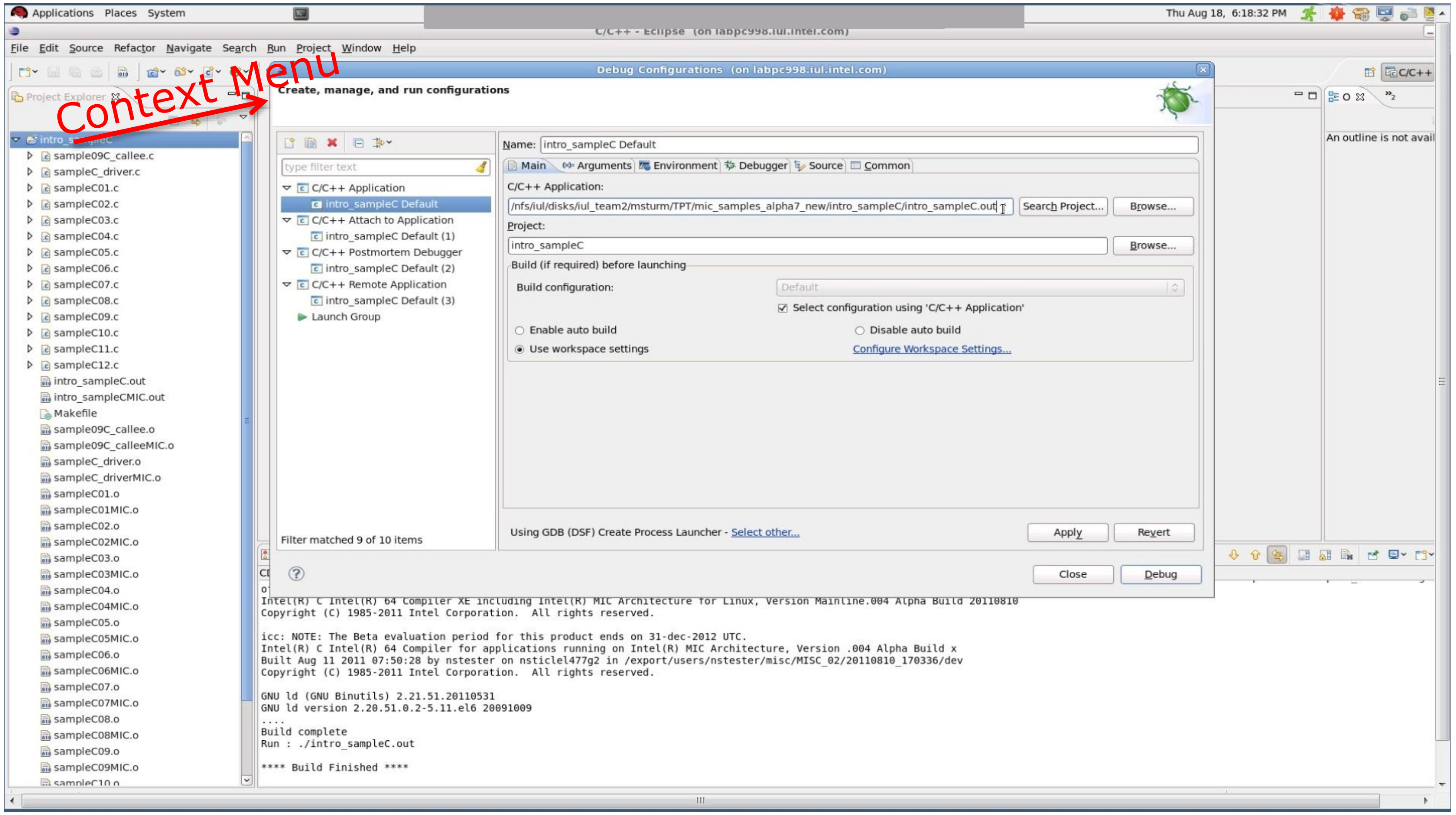
Description
Start new MIC application under control of IDB, integrated

Using GDB (DSF) Create Process Launcher - Select other...

Filter matched 9 of 10 items

Select Application to Debug

At the **Main** tab of the **Debug Configurations** dialog, enter the path to the application you wish to debug in the field **C/C++ Application**.


Context Menu
Debug Configurations (on labpc998.iul.intel.com)
Create, manage, and run configurations
C/C++ Application
intro_sampleC Default
C/C++ Attach to Application
intro_sampleC Default (1)
C/C++ Postmortem Debugger
intro_sampleC Default (2)
C/C++ Remote Application
intro_sampleC Default (3)
Launch Group
Main
Arguments
Environment
Debugger
Source
Common
Search Project...
Browse...
Apply
Revert
Close
Debug

Name: intro_sampleC Default

C/C++ Application: /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out

Project: intro_sampleC

Build (if required) before launching

Build configuration: Default

[✓] Select configuration using 'C/C++ Application'

( ) Enable auto build ( ) Disable auto build

(●) Use workspace settings Configure Workspace Settings...

Filter matched 9 of 10 items

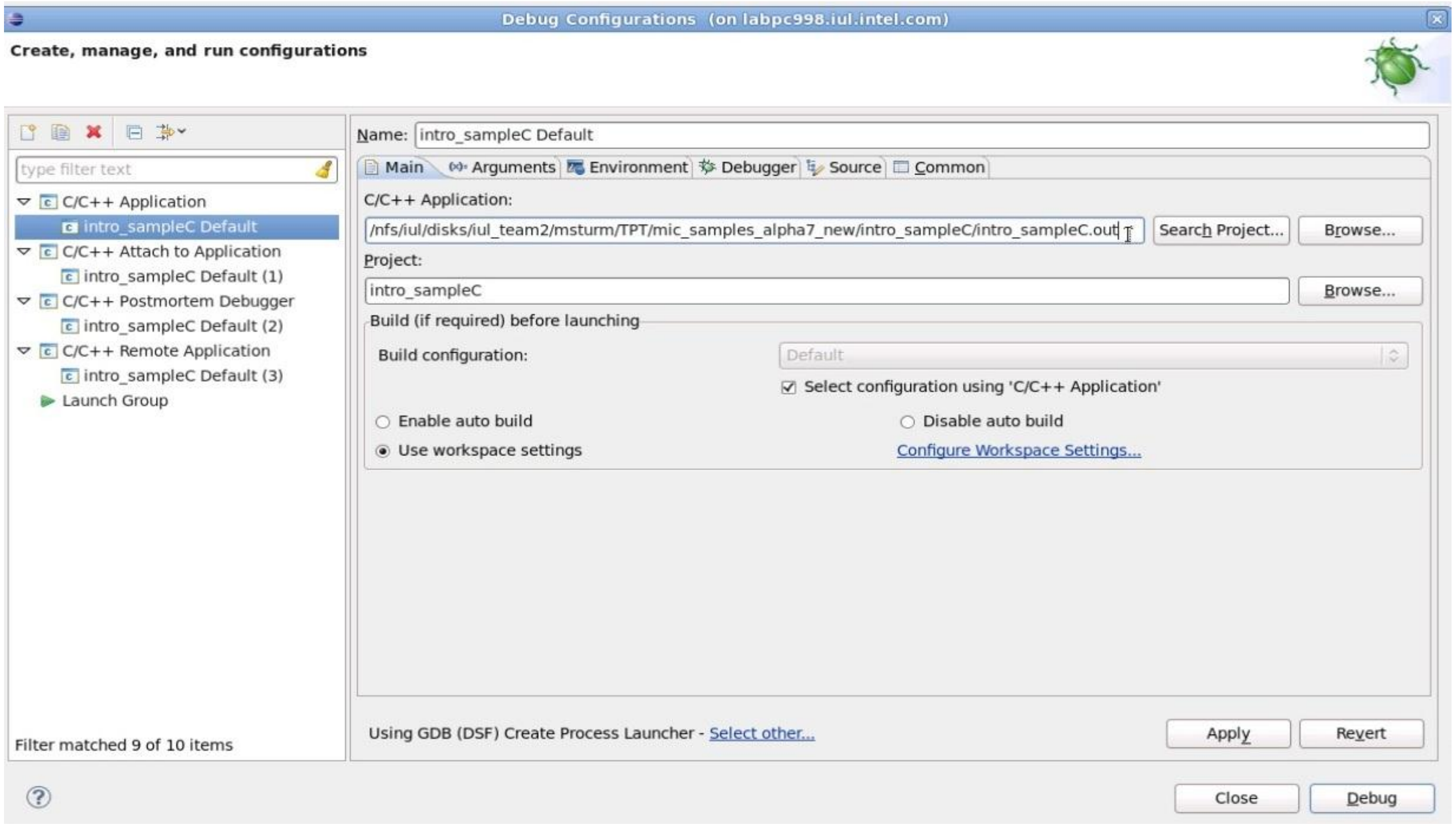
Using GDB (DSF) Create Process Launcher - Select other...

```
Intel(R) C Intel(R) 64 Compiler XE including Intel(R) MIC Architecture for Linux, Version Mainline.004 Alpha Build 20110810
Copyright (C) 1985-2011 Intel Corporation.  All rights reserved.

icc: NOTE: The Beta evaluation period for this product ends on 31-dec-2012 UTC.
Intel(R) C Intel(R) 64 Compiler for applications running on Intel(R) MIC Architecture, Version .004 Alpha Build x
Built Aug 11 2011 07:50:28 by nstester on nsticlel477g2 in /export/users/nstester/misc/MISC_02/20110810_170336/dev
Copyright (C) 1985-2011 Intel Corporation.  All rights reserved.

GNU ld (GNU Binutils) 2.21.51.20110531
GNU ld version 2.20.51.0.2-5.11.el6 20091009
....
Build complete
Run : ./intro_sampleC.out

**** Build Finished ****
```

Optimization Notice

Select Application to Debug

At the **Main** tab of the **Debug Configurations** dialog, enter the path to the application you wish to debug in the field **C/C++ Application**.


Debug Configurations (on labpc998.iul.intel.com)
Create, manage, and run configurations
type filter text
C/C++ Application
intro_sampleC Default
C/C++ Attach to Application
intro_sampleC Default (1)
C/C++ Postmortem Debugger
intro_sampleC Default (2)
C/C++ Remote Application
intro_sampleC Default (3)
Launch Group
Filter matched 9 of 10 items
Main
Arguments
Environment
Debugger
Source
Common
Search Project...
Browse...
Browse...
Configure Workspace Settings...
Select other...
Apply
Revert
Close
Debug

Name: intro_sampleC Default

C/C++ Application:
/nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out

Project:
intro_sampleC

Build (if required) before launching

Build configuration: Default

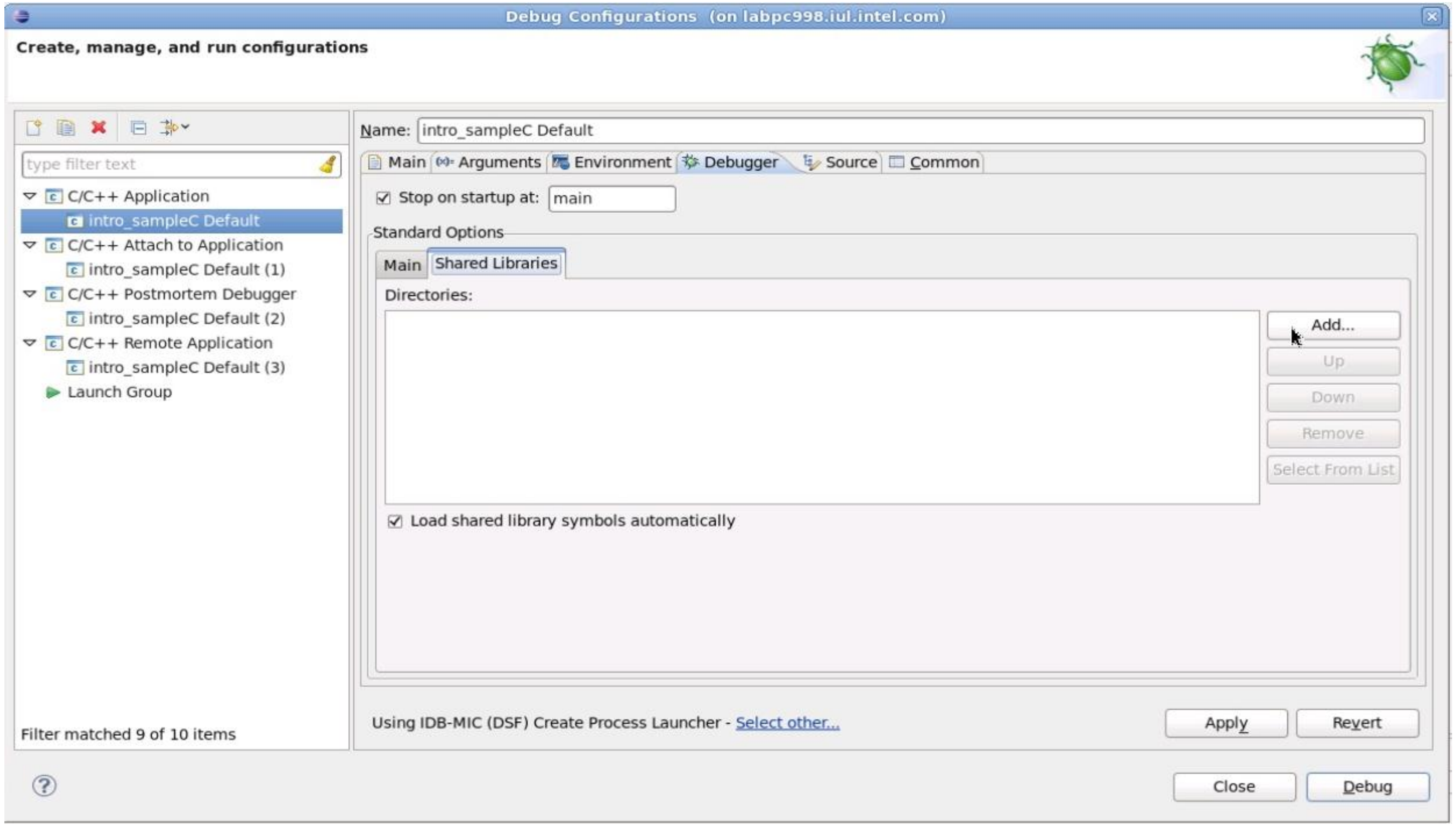
☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings

Using GDB (DSF) Create Process Launcher - Select other...

Add Search Directories for Shared Objects



Create, manage, and run configurations

Name: intro_sampleC Default

☑ Stop on startup at: main

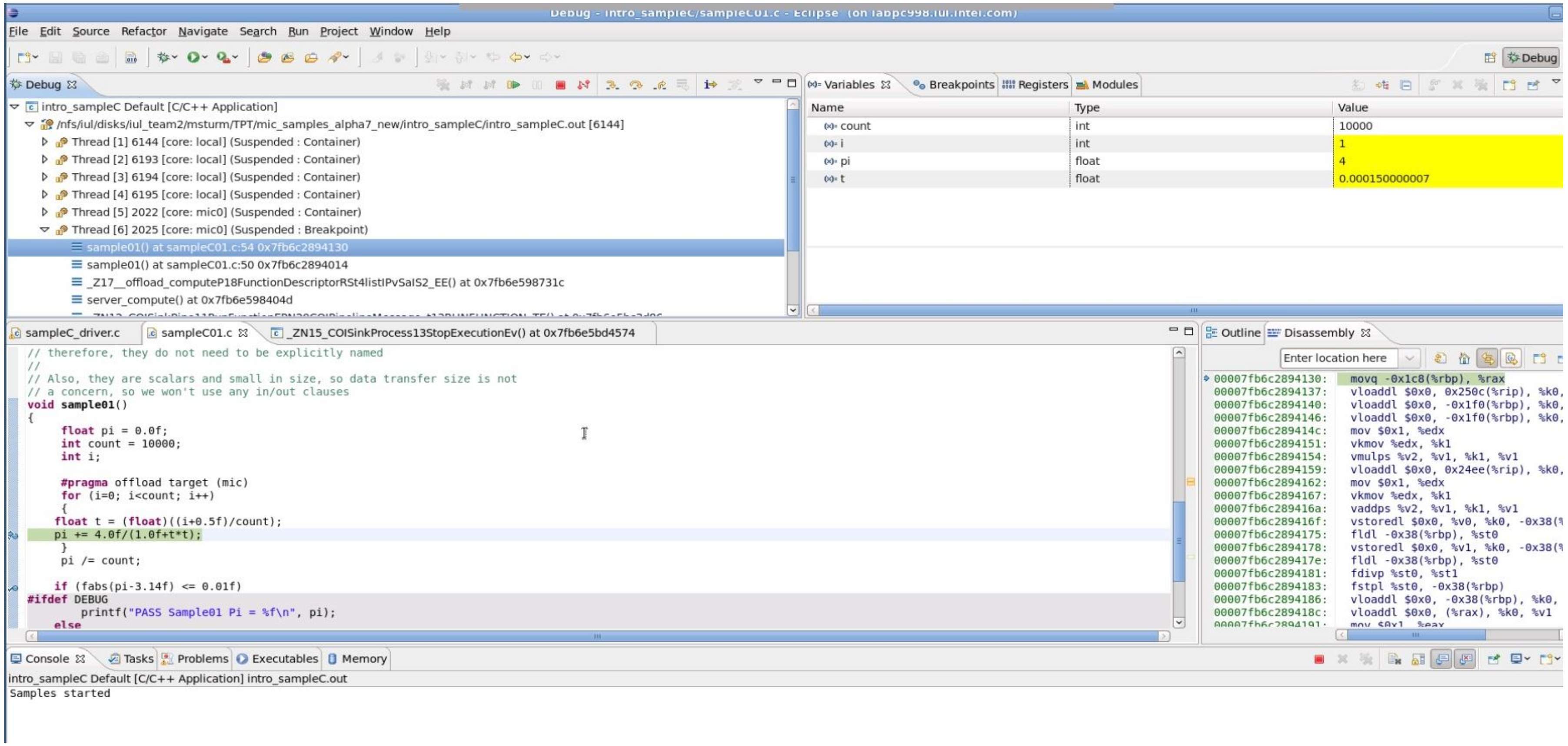
☑ Load shared library symbols automatically

Filter matched 9 of 10 items

Using IDB-MIC (DSF) Create Process Launcher - Select other...

Debug Offload Code from within Eclipse* Debug Perspective

Select **Run > Debug** from the menu bar or click the **Debug** button


Debug - intro_sampleC/sampleC01.c - Eclipse (on iabpc998.iul.intel.com)
File Edit Source Refactor Navigate Search Run Project Window Help
Debug

Debug

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

Variables | Breakpoints | Registers | Modules

Name	Type	Value
count	int	10000
i	int	1
pi	float	4
t	float	0.000150000007

sampleC_driver.c | sampleC01.c | _ZN15_COISinkProcess13StopExecutionEv() at 0x7fb6e5bd4574

```c
// therefore, they do not need to be explicitly named
//
// Also, they are scalars and small in size, so data transfer size is not
// a concern, so we won't use any in/out clauses
void sample01()
{
     float pi = 0.0f;
     int count = 10000;
     int i;

     #pragma offload target (mic)
     for (i=0; i<count; i++)
     {
    float t = (float)((i+0.5f)/count);
    pi += 4.0f/(1.0f+t*t);
     }
     pi /= count;

    if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
        printf("PASS Sample01 Pi = %f\n", pi);
    else
```

Outline | Disassembly

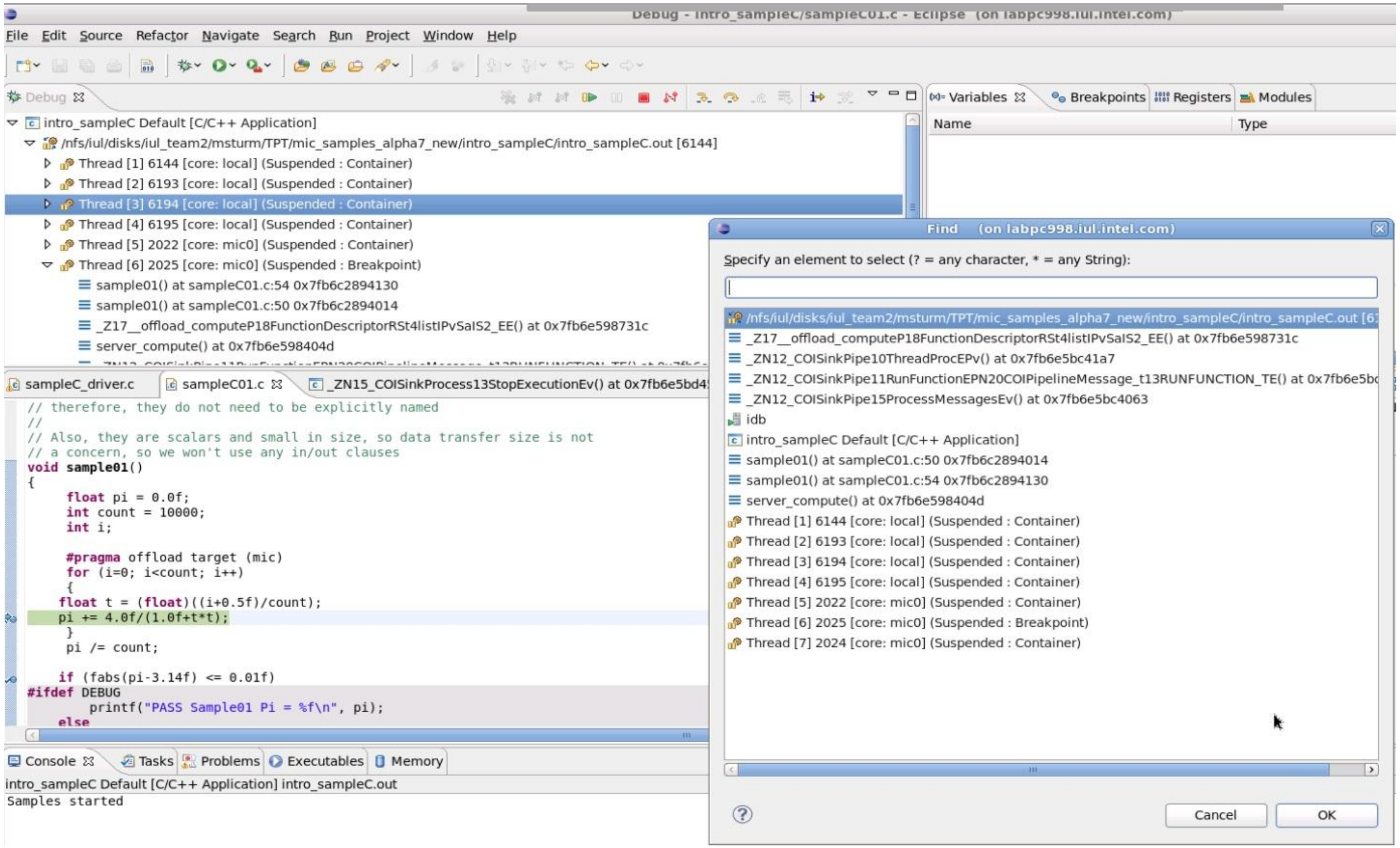
Enter location here

```
00007fb6c2894130:   movq -0x1c8(%rbp), %rax
00007fb6c2894137:   vloaddl $0x0, 0x250c(%rip), %k0,
00007fb6c2894140:   vloaddl $0x0, -0x1f0(%rbp), %k0,
00007fb6c2894146:   vloaddl $0x0, -0x1f0(%rbp), %k0,
00007fb6c289414c:   mov $0x1, %edx
00007fb6c2894151:   vkmov %edx, %k1
00007fb6c2894154:   vmulps %v2, %v1, %k1, %v1
00007fb6c2894159:   vloaddl $0x0, 0x24ee(%rip), %k0,
00007fb6c2894162:   mov $0x1, %edx
00007fb6c2894167:   vkmov %edx, %k1
00007fb6c289416a:   vaddps %v2, %v1, %k1, %v1
00007fb6c289416f:   vstoredl $0x0, %v0, %k0, -0x38(%
00007fb6c2894175:   fldl -0x38(%rbp), %st0
00007fb6c2894178:   vstoredl $0x0, %v1, %k0, -0x38(%
00007fb6c289417e:   fldl -0x38(%rbp), %st0
00007fb6c2894181:   fdivp %st0, %st1
00007fb6c2894183:   fstpl %st0, -0x38(%rbp)
00007fb6c2894186:   vloaddl $0x0, -0x38(%rbp), %k0,
00007fb6c289418c:   vloaddl $0x0, (%rax), %k0, %v1
```

Console | Tasks | Problems | Executables | Memory

```
intro_sampleC Default [C/C++ Application] intro_sampleC.out
Samples started
```

View Offload Threads



Debug - intro_sampleC/sampleC01.c - Eclipse (on labpc998.iul.intel.com)

File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Variables
Breakpoints
Registers
Modules
Name
Type

```
intro_sampleC Default [C/C++ Application]
  /nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6144]
    Thread [1] 6144 [core: local] (Suspended : Container)
    Thread [2] 6193 [core: local] (Suspended : Container)
    Thread [3] 6194 [core: local] (Suspended : Container)
    Thread [4] 6195 [core: local] (Suspended : Container)
    Thread [5] 2022 [core: mic0] (Suspended : Container)
    Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
      sample01() at sampleC01.c:54 0x7fb6c2894130
      sample01() at sampleC01.c:50 0x7fb6c2894014
      _Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
      server_compute() at 0x7fb6e598404d
```

```
// therefore, they do not need to be explicitly named
//
// Also, they are scalars and small in size, so data transfer size is not
// a concern, so we won't use any in/out clauses
void sample01()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    for (i=0; i<count; i++)
    {
    float t = (float)((i+0.5f)/count);
    pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;

   if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
       printf("PASS Sample01 Pi = %f\n", pi);
   else
```

```
intro_sampleC Default [C/C++ Application] intro_sampleC.out
Samples started
```

Find (on labpc998.iul.intel.com)

Specify an element to select (? = any character, * = any String):

```
/nfs/iul/disks/iul_team2/msturm/TPT/mic_samples_alpha7_new/intro_sampleC/intro_sampleC.out [6
_Z17__offload_computeP18FunctionDescriptorRSt4listIPvSaIS2_EE() at 0x7fb6e598731c
_ZN12_COISinkPipe10ThreadProcEPv() at 0x7fb6e5bc41a7
_ZN12_COISinkPipe11RunFunctionEPN20COIPipelineMessage_t13RUNFUNCTION_TE() at 0x7fb6e5b
_ZN12_COISinkPipe15ProcessMessagesEv() at 0x7fb6e5bc4063
idb
intro_sampleC Default [C/C++ Application]
sample01() at sampleC01.c:50 0x7fb6c2894014
sample01() at sampleC01.c:54 0x7fb6c2894130
server_compute() at 0x7fb6e598404d
Thread [1] 6144 [core: local] (Suspended : Container)
Thread [2] 6193 [core: local] (Suspended : Container)
Thread [3] 6194 [core: local] (Suspended : Container)
Thread [4] 6195 [core: local] (Suspended : Container)
Thread [5] 2022 [core: mic0] (Suspended : Container)
Thread [6] 2025 [core: mic0] (Suspended : Breakpoint)
Thread [7] 2024 [core: mic0] (Suspended : Container)
```

Optimization Notice

Debugging Only on the Coprocessor

1. Select **Run > Debug Configurations**....

2. Select the debug configuration type **C/C++ Attach to Application**.

3. Click the **New** button.

4. Enter a name for your configuration.

5. On the **Main** tab locate the process launcher information and click **Select other**....

The **Select Preferred Launcher** dialog box appears.

6. Check the **Use configuration specific settings** checkbox.

7. Select **IDB-MIC (DSF) Attach to Process Launcher**.

8. Click **OK**.

The **Select Preferred Launcher** dialog box is closed.

9. Switch to the **Debugger** tab.

10. Under **Options** check the **Attach to Intel® Xeon Phi™** checkbox.

11. Choose the desired coprocessor from the **Card** pull down menu.

12. At **File Location** browse for the symbol file related to the process you want to attach to.

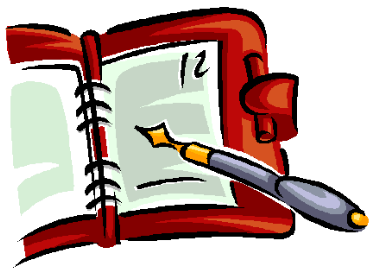
13. Click **Debug**.

The debugger starts and the **Select a Process** dialog box opens displaying a list of running processes.

14. Select the process you want to attach to and click **OK**.

Optimization Notice

Agenda



The GNU* Project Debugger and Intel® Xeon Phi™ Coprocessor

GDB* native-only debugger released.

(http://software.intel.com/en-us/forums/showthread.php?t=105443)

The GNU* Project Debugger and Intel® Xeon Phi™ Coprocessor

Modified and rebuilt GDB* 7.4

- Unpack

```
     tar xzf gdb-intel-mic-2.1.xxxx.tgz
```

- Copy to target

```
     export CARD=172.xxx.x.xxx

    scp gdb root@$CARD:/usr/bin
```

- Use GDB* locally in target processor terminal to attach and launch native process as you would on standard Linux*
- Sources to rebuild and modify GDB* and patches are provided.

More questions?

Optimization Notice