TotalView’s Support of Intel Xeon Phi 2

Debugging Offload Directives (LEO) 3

Debugging Native Applications 8
  Intel Xeon Phi Native Debugging 8
    Using a Standard Single Server Launch String 8
    Using the MIC Native Launch String 9
  Intel Xeon Phi MPI Native Debugging 10

Xeon Phi Options and State Variables 13
  Xeon Phi Command Line Options 13
  Zeon Phi-Specific State Variables 13

Known Issues 14
TotalView’s Support of Intel Xeon Phi

The Intel Xeon Phi coprocessor, based on Intel’s Many Integrated Core (MIC) architecture, is a major advancement in the performance and speed of parallel processing. Xeon Phi (and its current implementation codenamed “Knights Corner”), is designed for highly-parallel workloads and will scale to over 50 Intel architecture cores.

With the 8.12 release, TotalView now provides developers the ability to view, control, and debug codes running on both the host processor and the Intel Xeon Phi coprocessor. It supports host side applications using the Intel offload directives (LEO).

Developers can also debug applications running natively on the Phi coprocessor. Also supported is the ability to debug scalable MPI applications that are launched from the host environment, but run as native on one or more Intel Xeon Phi coprocessors on a server or across the nodes of an enabled cluster.
In this mode, the compiler allows a program to select computations to offload for running on specified hardware, in this case an MIC card. TotalView automatically detects offload runtime events and attaches the debugger to offload processes running on all cards.

In order to debug on the coprocessor, TotalView will launch a debug server on each card using the `micnativeloadex_server_launch_string`.

This should work in most cases, but if it fails, you can edit the `micnativeloadex_server_launch_string` variable, as discussed in “Using a Standard Single Server Launch String” on page 8.

1. Start TotalView on the host system as usual. For example:
   ```bash
totalview simple_offload
```
   TotalView loads your program but does not yet start it:
Set some breakpoints. You can set breakpoints in both host and offload code. For example, if you set a breakpoint at line 25 in offloaded code, this breakpoint will be hit either after offload occurs or, if the host has no available MIC cards, when the program running on the host reaches the breakpoint. Breakpoints outside offload programs are hit only by host processes running on the Xeon host.

After you set initial breakpoints, run your program.

Press the Go button.

Note that the console reports:

```
Launching TotalView Debugger Server with command:
micnativeloadex /nfs/toolworks/totalview.8X.12.0-0/linux-x86-64/bin/tvdsvrmain_mic -d 1 -a "-callback 172.28.29.11:16381 -set_pw 755f8be:7512e5cc -verbosity info -cuda"
```

The Intel tool is `micnativeloadex` installed in your system's `/opt/` directory, and the number after `-d` corresponds to the card number + 1. Is it the card number with 1 appended, or 1 added? Or is “+1” the card number?

If MIC cards are available in the system, the debugging process is similar to parallel debugging. After hitting the first offload directive, the Intel runtime launches a special program, `offload_main`, on the card and TotalView tries to attach to it.
If you have two cards in the system, select **Continue** and then **Go**. This dialog again launches when `offload_main` is started on the second card.

You can avoid these dialogs by setting **Preferences > Parallel** to attach automatically:
In this case, the debugger attaches to all cards and continues running to the first breakpoint.
You can also change the preference to attach to all processes and stop the group after each offload attach by selecting the **Stop the group** checkbox under "When a job goes parallel or calls exec()."
4 Hit **Go** again. TotalView runs until it hits the first breakpoint, Breakpoint 1 on line 25, in this case.

Since this is a multi-process, multi-threaded application, you can switch between different threads and processes using the usual controls in the Process or Root windows and debug in the same way as any other parallel and or multi-process applications.
Debugging Native Applications

**Intel Xeon Phi Native Debugging**

You can debug applications running natively on the Intel Xeon Phi coprocessor. As with the offload directives mode, TotalView launches its debug server on the coprocessor that will start the remote application.

There are two options for the launch:

- Using a standard single server launch string
- Using the MIC native launch string.

**Note:** This is the recommended method because it avoids having to explicitly change environment settings and preferences.

**Using a Standard Single Server Launch String**

This method can be useful if TotalView is installed on a file system that is accessible from all Xeon Phi coprocessors. In this case, you can use the standard server launch string, controlled by the `TV::server_launch_string` variable, which you can also set in UI preferences. Assuming that your program executable is also visible to both the coprocessor and the host, you can start debugging by just running

```
totalview -r host-micN mic_native_hello
```

where `host-micN` is the Xeon Phi coprocessor and `mic_native_hello` is the program to debug.

TotalView starts its debug server and displays the following message in the console:

```
Launching TotalView Debugger Server with command:
rsh host-mic0 -n "/nfs/toolworks/totalview.8X.12.0-0/linux-x86-64/bin/tvdsvr -working_directory <Current_dir> -callback 172.28.29.11:16381 -set_pw 755f8bfe:7512e5cc -verbosity info -cuda"
```

**Note that**

- The default remote launch command is `rsh`, so you need to change this to your system’s launch command — for example `ssh` — either in your Preferences or by setting `env TVDSVRLAUNCHCOMMAND=ssh`.
- The `tvdsvr` location (`/nfs/toolworks/totalview.8X.12.0-0/linux-x86-64/bin/` in this case) should be accessible from the coprocessor.
- The `<Current_dir>` path to your executable should be accessible as well. If it is not accessible, you need to change it or define it in your environment.

If your file system is different from that described, you can copy your executable and `tvdsvrmain_mic` to the location `/tmp/` on both the host and coprocessor and modify the launch string from the default

```
%C %R -n "%B/tvdsvr%K -working_directory %D -callback %L -set_pw %P %F"
```

to something like:
ssh %R -n "/tmp/tvdsvrmain_mic -working_directory /tmp/ -callback %L -set_pw %P %F"

For more information, see the TotalView 8.12 documentation, Chapter 18, "Setting Up Remote Debugging Sessions" in the TotalView User Guide.

After you save these changes, you may unable to perform remote debugging on a Linux-x86-64 remote machine without reverting the updates to the Xeon Phi launch string. For this reason, we recommend that you instead use the the special mic_native_launch_string together with -mmic option discussed below.

Using the MIC Native Launch String

This method is useful in two primary use cases:

- To perform debugging on both the host and Xeon Phi while maintaining separate server launch strings
- To run Totalview in an environment in which it is not installed on an accessible, shared file system. In this case, you can customize the TV::mic_native_launch_string in the global or personal .tvdrc file and start Totalview either with the -mmic flag for debugging on a Xeon Phi coprocessor, or without it for normal debugging.

For example:

dset TV::mic_native_server_launch_string { //1
    ssh -n %R "/bin/rm -f /tmp/tvdsvrmain%K"; //2
    scp %B/tvdsvrmain%K %R:/tmp/tvdsvrmain_mic; //3
    ssh -n %R -n "/tmp/tvdsvrmain%K -callback %L -set_pw %P -verbosity %V %F"
    }

- //1 Removes your previous tvdsvrmain_mic
- //2 Copies it from the installation directory to the /tmp/ directory on the coprocessor
- //3 Starts the server on the Xeon Phi coprocessor.

Now launch TotalView and start your remote debugging session using this launch string:

totalview -mmic -r host-micN mic_native_hello
Note that even though you started TotalView on the host, the debugged process is running on the coprocessor side (minnie-mic0, viewable above). Debug the executable in the same way you would debug an ordinary program on a remote host (CPU).

**Intel Xeon Phi MPI Native Debugging**

Debugging MPI parallel applications on an MIC card is no different than debugging parallel applications on a CPU-based cluster environment. You can also use two mechanisms to launch debug servers on the coprocessor, as described in “Intel Xeon Phi Native Debugging” on page 8.

However, you must use TotalView’s classic launch (that is, `mpiexec`). In addition to copying the standard MPI libraries to the MIC coprocessors, you need to copy the `libmpi_dbg.so` and `libmpi_dbg_mt.so` libraries as well. (If MPI was installed on an accessible, shared file system, you shouldn’t have this problem.)

To start debugging with Intel MPI, first verify that you can run your program without TotalView:

```
mpiexec -np 40 -host host@mic0 -wdir /tmp/ ./tx_basic_mpi
```

Then just add `totalview -args` before `mpiexec`, like this:

```
totalview -args mpiexec -np 40 -host root@mic0 -wdir /tmp/ ./tx_basic_mpi
```
or, to use the `mic_native_server_launch_string`, like so:

```
totalview -mmic -args mpiexec -np 40 -host root@mic0 -wdir /tmp/ ./
tax_basic_mpi if you want
```

You can also debug a multi-host, multi-card MPI job in the same way:

```
totalview -args mpiexec -np 5 -host host1-mic0 -wdir /tmp/ ./tax_basic_mpi
```
or

```
totalview -args mpiexec -np 250 -hosts host1-mic0,host1-mic1,host2-mic0,host2-mic1 ./tx_basic_mpi
```

If you are debugging a multi-host MPI application on Xeon Phi coprocessors, you need to satisfy following conditions:

- Each card must have its own IP address and be accessible from the front host node, running TotalView.
- TotalView must be installed in a global area and be accessible from each card in allocation, so that you can start `tvd mic-server` on each `mic-card` from the partition, or you can copy `tvdsvr` using the `mic_native_server_launch_string`. 

Note the Root Window when running on two MIC cards:

<table>
<thead>
<tr>
<th>ID</th>
<th>Rank</th>
<th>Host</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>&lt;local&gt;</td>
<td>T</td>
<td>(1 active threads)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexec (1 active threads)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0 mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;0 (1 active threads)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1 mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;1 (1 active threads)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2 mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;2 (1 active threads)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>3 mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;3 (1 active threads)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>4 mickey-mic0.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;4 (1 active threads)</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>(5 active threads)</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>5 mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;5 (1 active threads)</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>6 mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;6 (1 active threads)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>7 mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;7 (1 active threads)</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>8 mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;8 (1 active threads)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 mickey-mic1.totalviewtech.com</td>
<td>T</td>
<td>mplexecodeMpi_mic&gt;9 (1 active threads)</td>
</tr>
</tbody>
</table>
**Xeon Phi Options and State Variables**

### Xeon Phi Command Line Options

A new option has been added to the `totalview` command to support Xeon Phi.

- **-mmic**
  - Sets the remote system to Xeon Phi
  - Uses `mic_native_server_launch_string` instead of the single launch string
  - The option `-mmic` sets `TV::mic_native_launch` to `true` and selects the `TV::mic_native_server_launch_string` string to launch the `tvdvr`.
  - **Warning:** This option takes precedence over the `TV::micnativeloadex_server_launch_string`, so if you use offload and this option, then the `mic_native_server_launch_string` will be used instead of `micnativeloadex_server_launch`, and TotalView may not start.

### Xeon Phi-Specific State Variables

These new state variables support the use of Xeon Phi on TotalView.

- **TV::mic_native_launch** (true:false)
  - **Default:** `false`
  - When set to `true`, this is the same as using the `-mmic` option when starting TotalView.

- **TV::mic_native_server_launch_string**
  - **Default:** `%C %R -n "%B/tvdsvr%K -working_directory %D -callback %L -set_pw %P -verbosity %V %F"`
  - If `TV::mic_native_launch` is set to `true` or the `-mmic` command option is used, this variable defines the remote debug servers where TotalView will launch.

- **TV::micnativeloadex_server_launch_string**
  - **Default:** (if `micnativeloadex %B/tvdsvrmain_mic -d %d -a "-callback %L -set_pw %P -verbosity %V %F"`) Used for a Xeon Phi LEO (offload) launch, this launches the debug server on your Xeon Phi coprocessor using Intel’s `micnativeloadex` tool.
Known Issues

- ReplayEngine is not supported with Intel MIC.
- MemoryScape is not supported with Intel MIC.
- ZMM registers are shown as YMM and are not supported.