1. 概述

本手册描述了申威 26010 众核处理器 gdb 调试器的使用方式，目前支持申威 26010 程序调试的 gdb 有 2 个版本：一个是本地版，须登录到申威 26010 处理器上运行。另一个是远程版，在 x86 服务节点上运行，指定被调试的程序路径与被调试进程所在的计算节点号，对指定处理器节点进行远程调试。

在被调试程序通过 sw3run 或 bsub、srun 等命令加载运行的情况下，用户在调试过程中不再输入 run、rerun，需要先退出 quit，再启动。

运算核上的部分调试功能（watch、jump、return、finish）不支持，控制核支持 watch、jump、return、finish 调试命令。其余常用功能与 gnu gdb 7.2 兼容，支持 break, step, next, cont, print, info 等调试命令详见 gnu gdb 7.2 公开用户手册。

此外，调试对程序的编译有一定要求，如果需要打印变量，建议编译时一定要附加-g 编译。

2. 本地调试

本地调试有两种执行方式：

1） 通过 run 命令启动调试，调试退出后，在调试器中运行的程序也退出。

2） 指定正在运行的进程号，对程序进行动态调试，在调试退出后，程序继续运行。

1.1 例子程序

控制核：

```
#include <stdio.h>
#include "athread.h"

extern slave_AAA(void * arg);
main(int argc, char **argv)
{
        athread_init();
        athread_spawn(AAA, NULL);
        athread_join();
        athread_halt();
}
```

运算核：

```
….
18 AAA()
19 {
20     static __thread_local float xx[4] = { 0.321, 0.25};
21     xxx = 0xaa;
……
```

1.2 启动调试

为了简化步骤，要在启动调试的工作目录下，准备一个 gdbinit 文件，下面给出一个 gdbinit 文件示例。用户需要根据实际情况，修改该文件中 run 的参数、sym 的参数、运算核入口函数名，其它行不需要改动。

（一）gdbinit 文件内容示范如下：

```
file /sw3run
break   sys_m_run
run   -F 1 -i ./test        把 sw3run 的具体命令行参数加在 run 之后，
d 1
sym test        这里 test 表示 1.1 中例子程序编译产生的可执行文件名
break AAA       AAA 是运算核的入口函数。
cont
info thread
```

该 gdbinit 文件的作用是：先启动/sw3run 作为被调试程序，当/sw3run 执行到即将调度从核执行的时候，将众核程序作为新的被调试程序，使众核程序运行到运算核的入口函数。

（二）在控制核上运行 gdb

```
# gdb -x gdbinit
SW gdb (GDB) 7.2.7.5
Copyright (C) 2010 Free Software Foundation, Inc.
Contributed by JN. Computing Technology Institute in 2014

Breakpoint 1 at 0x1200003b8: file sw3arun.c, line 65.
[Thread debugging using libthread_db enabled]

Breakpoint 1, sys_m_run (_host_start=343333235232) at sw3arun.c:65
65    sw3arun.c: No such file or directory.
 in sw3arun.c
Slave Breakpoint 2 at 0x4ff0410d40: file ……/test/s.c, line 21.
  [Switching to slave 0]
AAA () at ……/test/s.c:21
21        xxx = 0xaa;
* 2 slave 0    AAA () at ……/test/s.c:21
    1 process 7587    __printf (format=0x4ff04b0640 "-----master spawn done--------\n") at
printf.c:30
(gdb)
```

至此，已到达运算核函数入口。

1.3 动态调试

```
# gdb -p 7787 _SELF
SW gdb (GDB) 7.2.7
Copyright (C) 2010 Free Software Foundation, Inc.
```

Contributed by JN. Computing Technology Institute in 2014

..

Reading symbols from…..1382/_SELF...done.

[Parallel-C debugging enabled]

Attaching to program: …..1382/_SELF, process 7787

[Thread debugging using libthread_db enabled]

0x0000004ff05841d0 in athread_join ()

(gdb) **i thr**

2 slave 0 0x0000004ff0411520 in _isaac64 () at …..1382/_1382.c:106

* 1 Thread 0x120122070 (LWP 7787) 0x0000004ff05841d0 in athread_join ()

(gdb)

(gdb) info cg **列出所有运算核的执行状态**

(gdb) t slave 1 **切换其他运算核，进行调试**

Thread 2 (slave 1):

[slave 1] #2 stopped.

0x0000004ff04115d0 in _isaac64 () at …..1382/_1382.c:107

107 rngstep(a ^ (a >> 33) , a, b, mm, m, m2, r, x);

(gdb) set slave-debugging together 设置为所有从核同行方式

在该方式下，执行 cont 命令时，所有从核都执行，如果命中从核断点，在全部从核都到达断点时，才打印(gdb)提示符，将执行控制返回给用户。

(gdb) set slave-debugging alone

设置从核独行方式(alone)：仅当前从核 cont，到达断点，就立即返回给用户查看，不等其它从核是否到达断点。

单步(step)命令总是当前核执行，其它核不执行。这一点与 cont 命令有所区别。

3. 远程调试

远程调试用于直接定位并行程序发生软件异常的位置。

被调试的并行程序在通 bsub 命令提交运行时，需要附加 –debug 参数，并且需要执行 bdebug 命令，启动远程节点上的调试服务程序，一个作业只要执行一次 bdebug 命令。

步骤示范如下：

$ export MV2_HANG_WHEN_ERROR=1 设置环境变量，当并行程序发生异常错误时，不退出，而是等待调速器来定位错误位置，如果要退出需要执行 bkill 命令。

进入 TEST 程序所在目录

$ bsub -debug -m 1 -I -b -p -q q_sw_yyz -host_stack 1024 -share_size 4096 -n 16 -cgsp 64 ./TEST

这里程序输出

CATCHSIG: Myid = 0(CPU 16541,CG 0), si_signo = 11(Segmentation Fault: PC = 0x4ff0a3d510)

提示程序在 16541 节点，发生 Segmentation Fault 错误，因此需要调试器进行错误定位。

步骤如下：

$ bjobs

JOBID	STAT	USER	JOB_NAME	QUEUE	FROM	SUBMIT_TIME	START_TIME	NODENUM	NODELIST
4375827	RUN	swpenv	TEST	q_sw_yyz	sn007	Jan 18 09:50	Jan 18 09:50	4	16541,16543,16545,16549

$ bdebug 4375827

Job <4375827> is being debugged

进入 TEST 程序所在目录，执行 swgdb

$ swgdb TEST 16541

SW gdb (GDB) 7.2.7.5

Copyright (C) 2010 Free Software Foundation, Inc.

Contributed by Parallel Development Labs in JN. Computing Technology Institute.

Released in 2015.

..

Reading symbols from …test/TEST/src/TEST...done.

task: 402 State: R, program: ./TEST

task: 403 State: R, program: ./TEST

task: 405 State: R, program: ./TEST

task: 407 State: R, program: ./TEST

(swgdb) attach 402

Attached to process 402

0x0000004ff07fb550 in .BB3_MPIDI_CH3_Abort (exit_code=..., error_msg=…) at src/mpid/ch3/channels/mrail/src/rdma/ch3_abort.c:30

(swgdb) where

#0 0x0000004ff07fb550 in .BB3_MPIDI_CH3_Abort (exit_code=..., error_msg=…) at src/mpid/ch3/channels/mrail/src/rdma/ch3_abort.c:30

#1 0x0000004ff07ea38c in .BB28_MPID_Abort (comm=..., mpi_errno=..., exit_code=..., error_msg=...) at src/mpid/ch3/src/mpid_abort.c:136

#2 0x0000004ff07aed98 in .BB32_PMPI_Abort (comm=..., errorcode=...) at src/mpi/init/abort.c:137

#3 0x0000004ff07ec1f0 in .BB14_catchit (i=..., info=..., ucontext=...) at src/mpid/ch3/src/mpid_init.c:672

#4 0x0000004ff097e860 in __syscall_rt_sigreturn ()

#5 0x0000004ff0a3d510 in ?? ()

#6 0x0000004ff0a09f34 in _IO_vfprintf_internal (s=..., format=…, ap=…) at vfprintf.c:1560

#7 0x0000004ff09fbef0 in __nldbl_vfprintf (s=..., fmt=…, ap=…) at ../sysdeps/ieee754/ldbl-opt/nldbl-compat.c:164

#8 0x0000004ff09fc034 in __nldbl_fprintf (stream=..., fmt=…) at ../sysdeps/ieee754/ldbl-opt/nldbl-compat.c:107

#9 0x0000004ff063ad3c in mpiPi_print_report_header (fp=..., report_style=...) at report.c:374

#10 mpiPi_profile_print (fp=..., report_style=...) at report.c:3036

#11 0x0000004ff06320c0 in mpiPi_publishResults (report_style=...) at mpiPi.c:954

#12 0x0000004ff0633300 in mpiPi_generateReport (report_style=...) at mpiPi.c:1105

---Type <return> to continue, or q <return> to quit---q

Quit

(swgdb) frame 9

#9 0x0000004ff063ad3c in mpiPi_print_report_header (fp=..., report_style=...) at report.c:374

(swgdb) p i

$1 = 36

(swgdb) p mpiPi.av[i]

$3 = 0x312e3230342e3631 <Address 0x312e3230342e3631 out of bounds>

这里就找到了程序出错的位置 report.c:374，由于 mpiPi.av[i]的地址非法导致访问内存错误。

4. 常用调试命令

命 令	缩写	用 法	作 用
help	h	h command	显示命令的帮助
step	s	s [n]	当前核步进,n 为步进次数。如果调用了某个函数，会跳入函数内部。
next	n	n [n]	下一步,n 为下一步的次数
cont	c	c	继续执行程序
list	l	l / l+ / l-	列出源码, list 和+、-之间有空格
break	b	b *address	在地址 address 上设置断点， address 前面有*字符
		b function	此命令用来在某个函数上设置断点。
		b linenum	在行号为 linenum 的行上设置断点。程序在运行到此行之前停止
		b +offset b -offset	在当前程序运行到的前几行或后几行设置断点。offset 为行号
		b filename:linenum	在文件名为 filename 的原文件的第 linenum 行设置断点
		b filename:function	在文件名为 filename 的原文件的名为 function 的函数上设置断点。当你的多个文件中可能含有相同的函数名时必须给出文件名
watch	wa	wa 变量名	监视变量的值，非寄存器，当变量的值发生改变时，停止程序的执行，并报告改变变量的语句或指令。
kill	k	k	结束当前调试的程序
print	p	p exp	打印表达式的值
		p/fmt exp	fmt 为以下值 x 十六进制 d 十进制 u 无符号数 o 八进制 t 二进制 a 十六进制打印 c 字符格式 f 浮点数
		p $a0， p $pc,p $sp p $f0， p $fpcr p $V0	打印寄存器
		p exp=val p $V0.intv8={0,1,2…} p $V0.doublev4={0.1,…}	给变量、寄存器赋值

ptype		ptype struct	输出一个 struct 结构的定义
whatis		whatis var	命令可以显示某个变量的类型
x	x/(length)(format)(size) addr 类似于 x/6(o/d/x/u/c/t)(b/h/w) 例如：x/4xw		按一定格式显示内存地址或变量的值，参见 fmt
pwd		pwd	显示当前路径
delete	d	d num	删除编号为 num 的断点和监视
disable		disable n	编号为 n 的断点暂时无效
enable		enable n	与 disable 相反
display		display expr display/fmt expr	暂停，步进时自动显示表达式的值
finish	fini		继续执行直到函数返回显示其返回值（如果有的话），适于控制核
return			强制从当前函数返回，跳过函数中未执行的语句，适于控制核
where	bt		显示函数调用的 踪迹。
quit	q		退出调试程序
frame	f	frame n	指定 where 中的某层函数，便于打印该函数中的局部变量
shell		shell ls	执行 shell 命令
disassemble	disass		显示反汇编代码
thread	**thr**	**thread 线程号**	**切换当前被调试线程，线程号见 info thread 命令的输出**
thread	**thr**	**thr slave 从核号**	**切换当前被调试从核，从核号见 info cg 命令的输出**
set		set width 70	就是把标准屏幕设为 70 列
		set var=54	设置变量的值。
up/down			上移/下移栈帧，使另一函数成为当前函数
info	i	i breakpoint	显示当前断点列表
		info reg/float/vec	显示寄存器组的当前值，通用/浮点/向量
		i threads	**显示线程信息**
		i source	查看当前源程序
		i func	显示所有的函数名
		i local	显示当前函数的所有局部变量的信息
		i prog	显示调试程序的执行状态
		i cg	**列出所有从核上程序运行的当前行。**
		i proc	显示进程的概要信息
		info proc mappings	报告你进程所能访问的地址范围。
		info proc times	你进程和子进程的开始时间，用户时间(user CPU time),和系统 CPU 时间。

		info proc id	报告有关进程 id 的信息
		info proc status	报告你进程的一般状态信息。如果进程停止了。这个报告还包括停止的原因和收到的信号
		info proc all	显示上面 proc 命令这些命令返回的所有信息
call		call func	调用和执行一个函数， 函数表达式包括参数和左右括号。
handle		handle signal action	Signal 为信号，如：SIGTRAP、SIGINT Action 为下列动作中一个或多个： Stop| nostop, print| noprint,pass|nopass, ignore|noignore

5. 功能缺陷

调试工具与编译器是紧密相关的，如果优化编译的目标文件及其依赖库中存在调试器无法理解的符号表信息条目，则调试器会报错“Dwarf Error”，则建议先退出调试，取消优化编译，采用-g –O0 编译程序，如果仍然未解决“Dwarf Error”问题，则联系相关技术人员。