Debugging
at SHARCNET

General Interest Webinar
18 06 2014

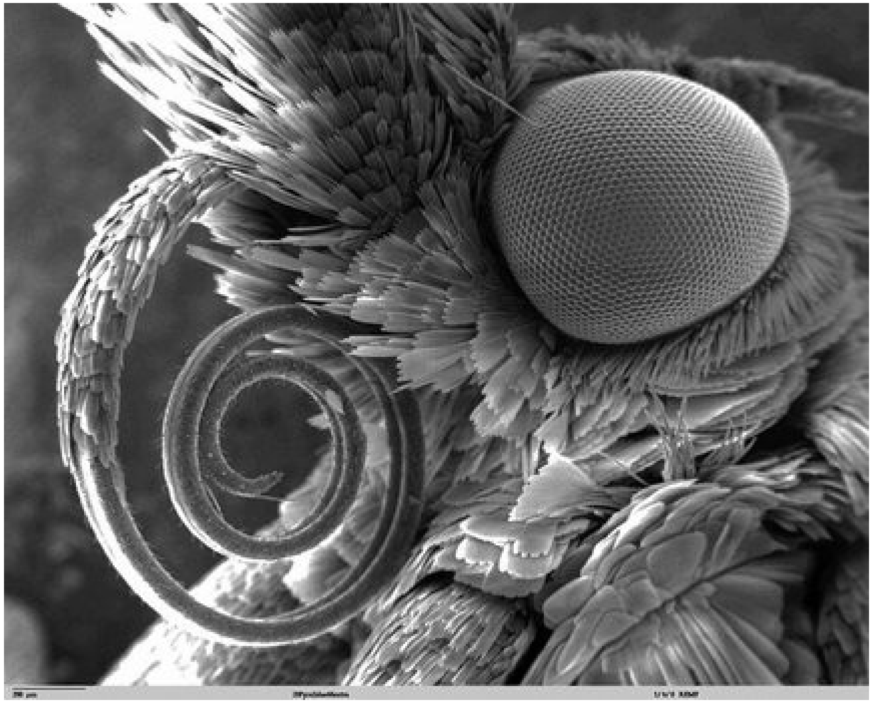
Hugh Merz

Session Outline

- How to diagnose job or program failures on SHARCNET systems
- How to identify and correct common programming bugs
- The use of *gdb* for debugging serial programs
- The use of *DDT* for debugging parallel programs

Identifying Bugs and Errors

- Typical signs that your program is buggy include:



  - It fails to complete (*crashes*)
  - It produces incorrect output (!##%?)
  - It fails to progress (*hangs*)

Diagnosing Job Failures

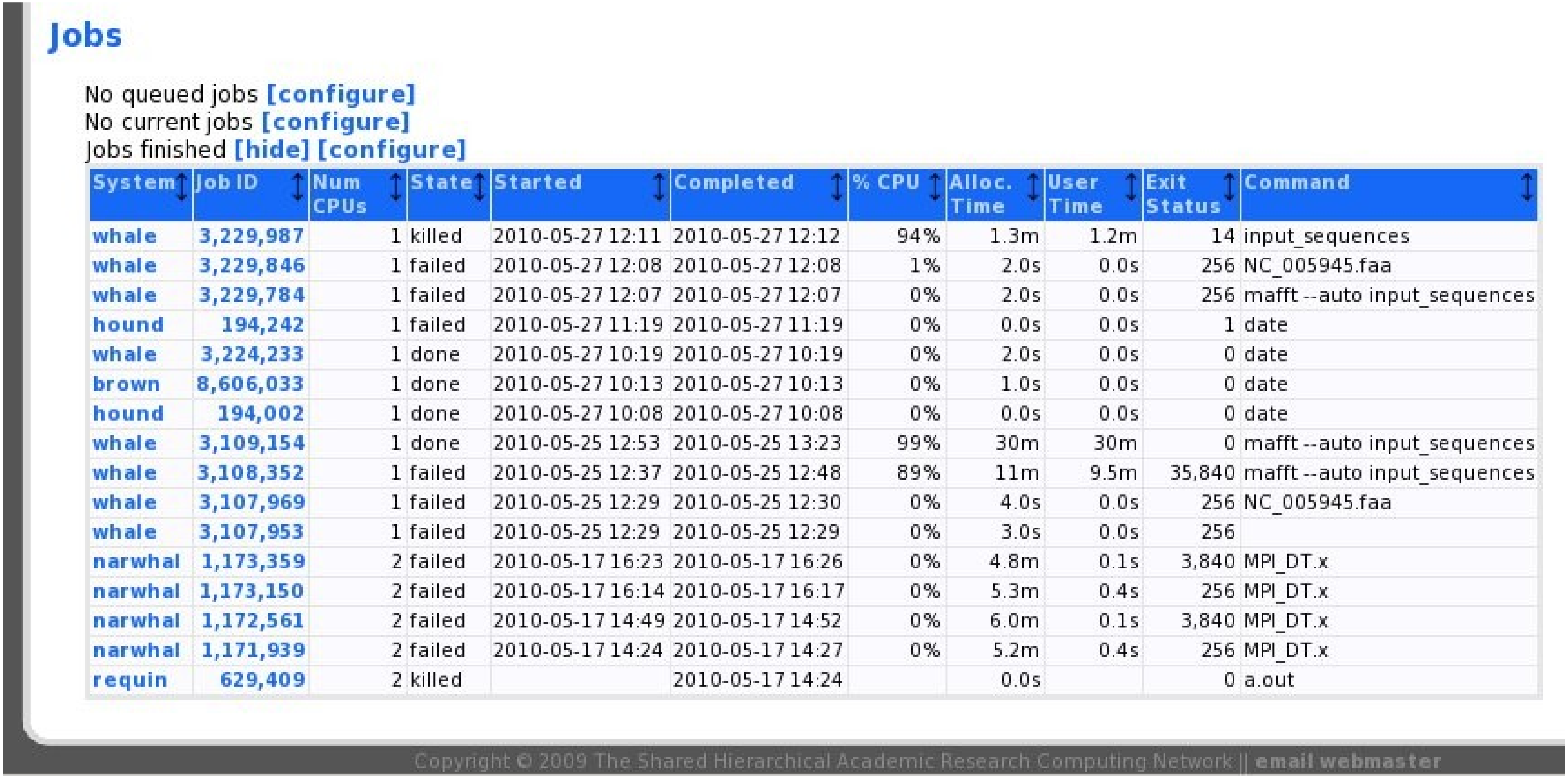
- Job (process) exit status (code)
- Job Scheduling and output handling
  - LSF vs. Torque/Maui/Moab , SQ
- Job Identifier (*jobid*)
- Web Portal Jobs Table
- The system's view of a job:
  - *sqjobs -l || bhist || qstat*
- Inspecting running jobs
  - *sqjobs -L* || SHARCNET Ganglia

Common Error Signals

Signal Name	OS signal #	OS signal name	Description
Floating point exception	8	SIGFPE	The program attempted an arithmetic operation with values that do not make sense (eg. divide by zero)
Segmentation fault	11	SIGSEGV	The program accessed memory incorrectly (eg. accessing an array beyond it's declared bounds)
Aborted	6	SIGABRT	Generated by the runtime library of the program or a library it uses, after having detected a failure condition
Kill	9	SIGKILL	The job management system terminates a job when it exceeds resource limits (eg. Runtime or memory)

Note: job exit status is typically 0 (success), negative (system issue) or `128+”OS signal #”` when there is a failure

Web Portal Jobs Table

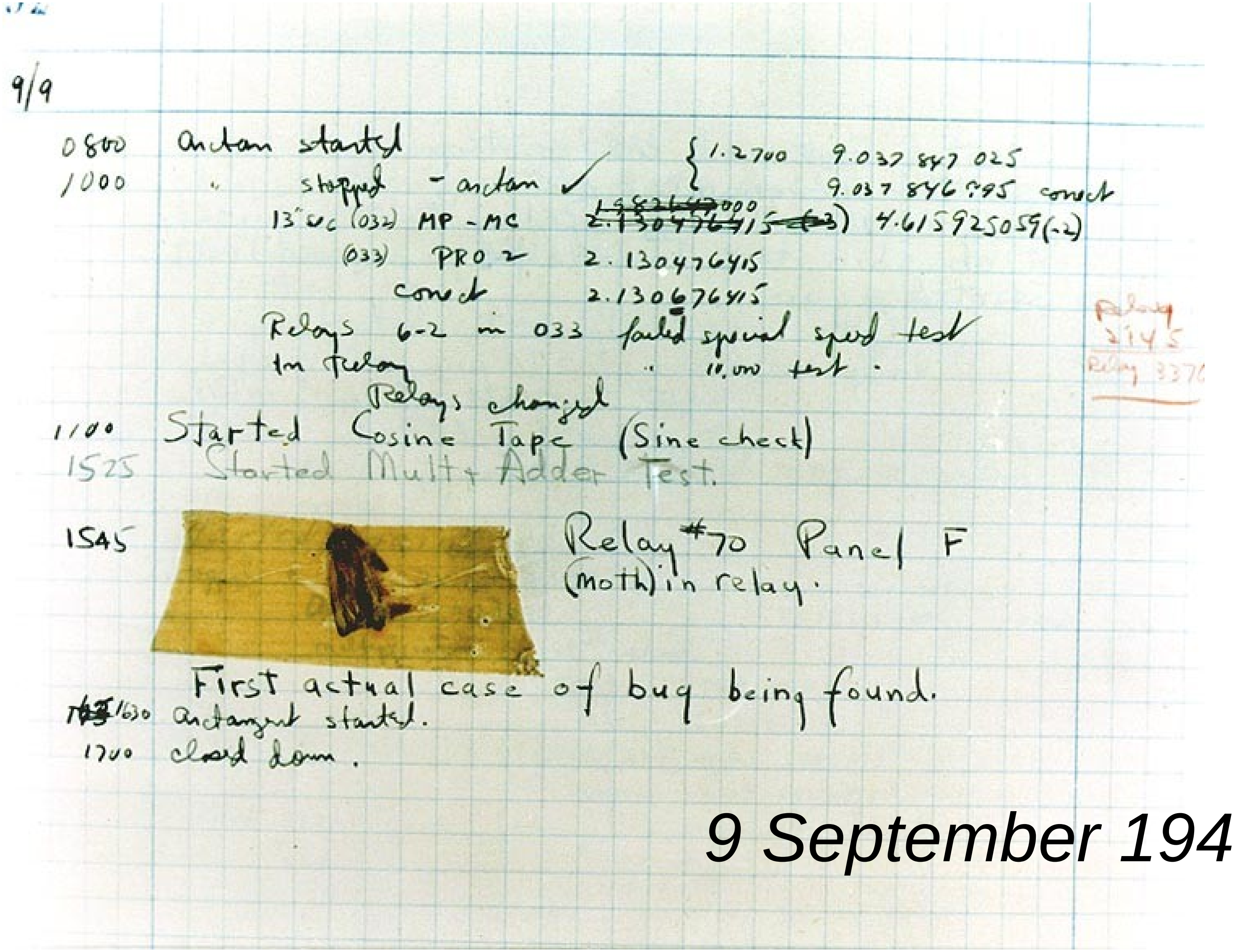


Jobs

No queued jobs [configure]
No current jobs [configure]
Jobs finished [hide] [configure]

System	Job ID	Num CPUs	State	Started	Completed	% CPU	Alloc. Time	User Time	Exit Status	Command
whale	3,229,987	1	killed	2010-05-27 12:11	2010-05-27 12:12	94%	1.3m	1.2m	14	input_sequences
whale	3,229,846	1	failed	2010-05-27 12:08	2010-05-27 12:08	1%	2.0s	0.0s	256	NC_005945.faa
whale	3,229,784	1	failed	2010-05-27 12:07	2010-05-27 12:07	0%	2.0s	0.0s	256	mafft --auto input_sequences
hound	194,242	1	failed	2010-05-27 11:19	2010-05-27 11:19	0%	0.0s	0.0s	1	date
whale	3,224,233	1	done	2010-05-27 10:19	2010-05-27 10:19	0%	2.0s	0.0s	0	date
brown	8,606,033	1	done	2010-05-27 10:13	2010-05-27 10:13	0%	1.0s	0.0s	0	date
hound	194,002	1	done	2010-05-27 10:08	2010-05-27 10:08	0%	0.0s	0.0s	0	date
whale	3,109,154	1	done	2010-05-25 12:53	2010-05-25 13:23	99%	30m	30m	0	mafft --auto input_sequences
whale	3,108,352	1	failed	2010-05-25 12:37	2010-05-25 12:48	89%	11m	9.5m	35,840	mafft --auto input_sequences
whale	3,107,969	1	failed	2010-05-25 12:29	2010-05-25 12:30	0%	4.0s	0.0s	256	NC_005945.faa
whale	3,107,953	1	failed	2010-05-25 12:29	2010-05-25 12:29	0%	3.0s	0.0s	256	
narwhal	1,173,359	2	failed	2010-05-17 16:23	2010-05-17 16:26	0%	4.8m	0.1s	3,840	MPI_DT.x
narwhal	1,173,150	2	failed	2010-05-17 16:14	2010-05-17 16:17	0%	5.3m	0.4s	256	MPI_DT.x
narwhal	1,172,561	2	failed	2010-05-17 14:49	2010-05-17 14:52	0%	6.0m	0.1s	3,840	MPI_DT.x
narwhal	1,171,939	2	failed	2010-05-17 14:24	2010-05-17 14:27	0%	5.2m	0.4s	256	MPI_DT.x
requin	629,409	2	killed		2010-05-17 14:24		0.0s		0	a.out

The First "Bug"



9/9

0800 Anctan started

1000 " stopped - anctan ✓ {1.2700 9.037 847 025

9.037 846 995 conect

13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415 (-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

conect 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F

(moth) in relay.

First actual case of bug being found.

~~143~~ 1630 Antangent started.

1700 closed down.

Relay 2145

Relay 3370

9 September 1947

Diagnosing the situation

- pay attention to compiler warnings
- inspect the job exit code in the web portal
- look at the job output file
    - may indicate a problem with the state of the program or a lack of progress
    - may contain a runtime error message, signal from the operating system or error from the job management system that helps identify the problem

Common Bugs

- Arithmetic
  - infinities, out of range
- Logic
  - infinite loop
- Syntax
  - wrong operator, arguments
- Resource starvation
  - memory leak
- Parallel
  - race conditions
  - deadlock
- Misuse
  - wrong initial conditions / insufficient checking / variable initialization

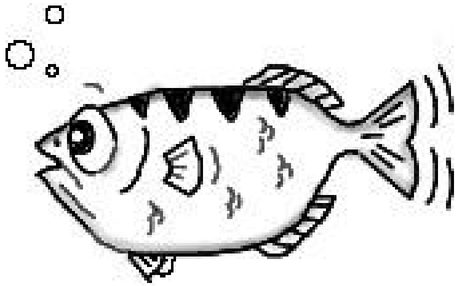
Floating Point Exceptions

- compilers/runtimes handle floating point exceptions differently
  - Some allow turning this on/off during compilation
  - Pathscale: -TENV:simd_*mask=OFF
  - Intel (fortran only!): -fpe0
  - gfortran: -ffpe-trap=invalid,zero,overflow
- Can also trap exceptions via library functions
  - glibc: feenableexcept()
    - compile and link to *trapfpe.c* code

Correcting Bugs

- if no error message is generated or if the message is insufficient to identify the problematic code one can use a **debugger**
- A debugger is a program that allows one to manipulate and inspect a second program as it is running
- Typically, the program should be compiled to include a symbol table (often **-g**) if you are going to run it in a debugger

gdb



- GNU Project Debugger
- Freely available, runs on most *nix systems, open source
- Works with a wide variety of languages
- Demonstration loosely following this tutorial in our help wiki:
  - Using gdb in the Online Training Centre

Debugging tips

- If your bug isn't repeatable:
  - Race condition? Randomness?
  - If a bug only appears with certain configurations / initial conditions it may be due to resource starvation or incorrect usage
- When reporting problems with the underlying system/software, provide a simple (and quick) test case, if possible
- Incorrect validation of input can result in many different errors!

Debugging tips

- Most Fortran compilers support runtime checking for out-of-bound array accesses, eg.
  - $ f90 -ffortran-bounds-check outbounds.f90
- Ensure that variables are defined with sufficient precision (overflow/underflow)
- Some MPIs support reporting more diagnostic information (eg. linking with hp-mpi's *-ldmpi*)
- Functionality in SHARCNET job submission to automatically generate a backtrace (LSF only):
  - *sqsub –backtrace ...*

DDT

- In addition to features in gdb:
  - GUI debugger (tabbed interface)
  - Shows multiple source files w/ syntax highlighting
  - Support for MPI, threaded and GPGPU debugging
    - Independant and group process/thread control (breakpoints, syncronization, comparisons)
  - inspection of variables (visualization, watches, checking pointers)
  - Visualization of MPI message queues
  - memory debugging!