Using MRNet and StackwalkerAPI to Deliver Scalable Analysis of Crashing Applications on Cray XT Systems

OR

Abnormal Termination Processing (ATP)

Bob Moench
Applications on Cray systems use hundreds of thousands of processes.
On a crash one, many, or all of them might trap.
No one wants that many core files.
No one wants that many stack backtraces.
They are too slow and too big.
They are too much to comprehend.
ATP Description

- System of light weight back-end monitor processes on compute nodes
- Coupled together with MRNet
- Leap into action on any application process trapping
- STAT like analysis provides merged stack backtrace tree
- Leaf nodes of tree define a modest set of processes to core dump
- Or, a set of processes to attach to with a debugger
STAT (Stack Trace Analysis Tool)

- Lawrence Livermore and University of Wisconsin
- Scalable collection of stack backtraces
- Fast, scalable, and compact
ATP Components

- Application process signal handler
  - triggers analysis
  - controls its own RLIMIT_CORE and core_pattern
- Back-end monitor
  - collects backtraces via StackwalkerAPI
  - forces core dumps as directed
- Front-end controller
  - coordinates analysis via MRNet
  - Selects process set that is to dump core
ATP Communications Tree

Front-end

Back-end

CP
BE
App

BE
App

BE
App

BE
App

BE
App

BE
App

BE
App

BE
App
ATP Back-end Interactions

ATP back-end

Main
Select()

MRNet

Signal Handler

fd

Socket

ATP back-end

App rank 1
ATP
Sig Handler

App rank 2
ATP
Sig Handler

App rank 3
ATP
Sig Handler

App rank 4
ATP
Sig Handler

Compute Node

PID

SIGUSR
MRNet Streams

- Control: Sends commands
  - TFILTER_SUM
  - SFILTER_WAITFORALL
- Crash: Request for ATP processing
  - TFILTER_SUM
  - SFILTER_DONTWAIT
- Backtrace: Delivery and merging of backtraces
  - TFILTER_Merged_Backtrace
  - SFILTER_WAITFORALL
ATP Requirements

- Minimum jitter
- Scalability
- Robustness
- Small footprint
- Limited core file dumping
- On by default
Limiting Core File Dumping

- ATP must overtly request dumping
- RLIMIT_CORE used to block accidental cascade of dumps
- core_pattern enhancement for "just in time" control of naming
Signal Handler Robustness

- Contrasted against ptrace
- Pre-allocated alternate stack
- State kept in read only memory segments
Additional features

- E-mail of failure status
- Stack backtrace of "first" failure to stderr
- List of signaled processes and their signal
- Checkpointable/restartable