DySectAPI: Scalable Prescriptive Debugging

Jensen, Nicklas Bo; Karlsson, Sven; Quarfot Nielsen, Niklas; Lee, Gregory L.; Ahn, Dong H.; Legendre, Matthew; Schulz, Martin

Publication date: 2014

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
DySectAPI: Scalable Prescriptive Debugging

Enabling users to construct probe trees for automatic, event-driven debugging at scale

Nicklas Bo Jensen, Sven Karlsson @ DTU Compute, Technical University of Denmark
Nicholas Quarfot Nielsen @ Mesosphere Inc
Gregory L. Lee, Dong H. Ahn, Matthew Legendre, Martin Schulz @ Computation, Lawrence Livermore National Laboratory

Motivation

- Current parallel debugging models are largely inadequate for extreme-scale systems
- Traditional model quickly overwhelms developers even at moderate scale
- Lightweight model, where information is sacrificed for scalability, scales well but can significantly limit debug information
- We need a new debugging model that can scale while still providing affordable information levels

Anatomy of a Debugging Probe

Events trigger probes. An event can either be a source code location or a signal.

Must be satisfied for the actions to be taken

Event: signal(SIGSEGV)
Condition: x > 2
Domain: group(1-512, 713, 715, timeout)
Actions: trace("Error at @function()")

Defines set of tasks in which to install this probe

Defines the wait time before aggregating the actions across the domain

Probe actions can be formulated as aggregation or reduction (e.g., aggregated messages, min/max, stack traces via the Stack Trace Analysis Tool, instruction traces\(\text{counts}\))

Probe Tree for a Real Use Case

When at each breakpoint, install the child probe in the triggered processes for filtering

Demonstrating Scalability

- Goal: show the scalability of our prescriptive debugging model
- Uses the STAT infrastructure, including MRNet, for scalable tool communication and data processing
- Use an analytic runtime-overhead model to predict scalability at large scale
- Logarithmic scaling demonstrated experimentally and modeled
- Our overhead model predicts a four probe chain-shaped tree would incur 118 ms in debugging the entire Cab system (20,736 cores)
- Pruning debug information via a probe tree significantly reduces information presented to programmers
- Four chained probes and pruning of 50% of the processes in each probe leave just 12.5% of the original processes

Impacts

- A novel debugging model that can scale without sacrificing key debug information presented to programmers
- Already proven effective in isolating an MPI bug that manifested itself only at or above 3,456 processes
- Debugging of common debugging scenarios using probe-tree templates
- Future work includes advanced probes using dynamic binary instrumentation

Source Code

Open source code available at: https://github.com/scalability-llnl/DysectAPI

Contributions

- A novel debugging model that can strike a balance between scalability and capability
- A prescriptive model
- Allows programmers to codify their debug intuition
- DySectAPI, an implementation of this model
- Performance and scalability evaluation
- Case study to demonstrate DySectAPI’s effectiveness

Figure: Debugging scalability based on strategies and amounts of details.

Figure: Enabling users to construct probe trees for automatic, event-driven debugging at scale.