Support in situ data processing on a supercomputer

Dr. Minh Ngoc Dinh
Problem statement

> I/O volume (e.g. data storage and data movement) is a critical bottleneck at extreme scale

> Traditional post processing workflow is time consuming
Research focus

> Large scale comparative debugging
  > Why? Large codes are constantly evolving
    > User requirements
    > Underlying algorithms,
    > Computer architecture.
  > Subtle data errors motivates data-centric techniques
    > Programmers can spend much time
    > Comparing the executions of two codes
    > Identify the source of a small discrepancy

> Runtime verification
  > Verifying large-scale scientific codes using robust statistical profiles
  > Measure the difference in data distribution to detect soft errors in the data
### Implementation Techniques

> **Assertion Engine**
> - Programs/processes are running asynchronously
> - Multiple assertions, can share same line numbers or variables
> - Assertions specify breakpoint locations in processes
> - Comparison process needs to be automated
> - Stop execution when threshold reached

> **Parallel Data Decomposition**
> - Assertions require the debugger to understand data decomposition
> - Currently support HPF style decompositions

> **Dataflow Engine**
> - Supports asynchronous behavior in debugged processes
> - Flexible assertion structure
> - Single program assertions
> - Cross coupled assertions
> - Multi-process parallel programs

```plaintext
blockmap test(P::V) 
define distribute(block, *)
  define data(1024, 1024)
end
```
Outcomes

> Guard – an open source relative debugger

> Cray CCDB and LGDB