



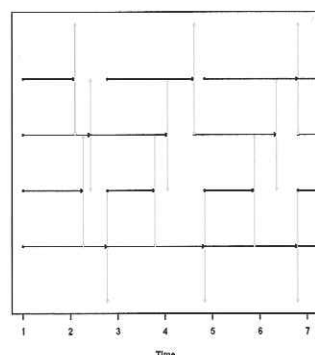
# Speed, Heat, and Noise

Stephanie Labasan\*, Barry Rountree†, Marc Casas Guix\*

University of Oregon\*, Center for Applied Scientific Computing\*†, Barcelona Supercomputing Center\*



In order to address how particular applications will execute on future architectures, we will need to model user defined noise patterns.  
This work will demonstrate the feasibility of a scalable noise model and simulation targeted for exascale machines.

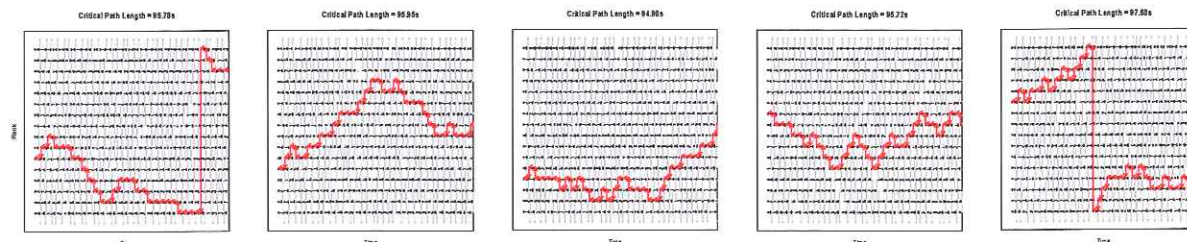


$$t_1: G(x) = x^3 \quad g(x) = \frac{dG}{dx} = 3x^2$$

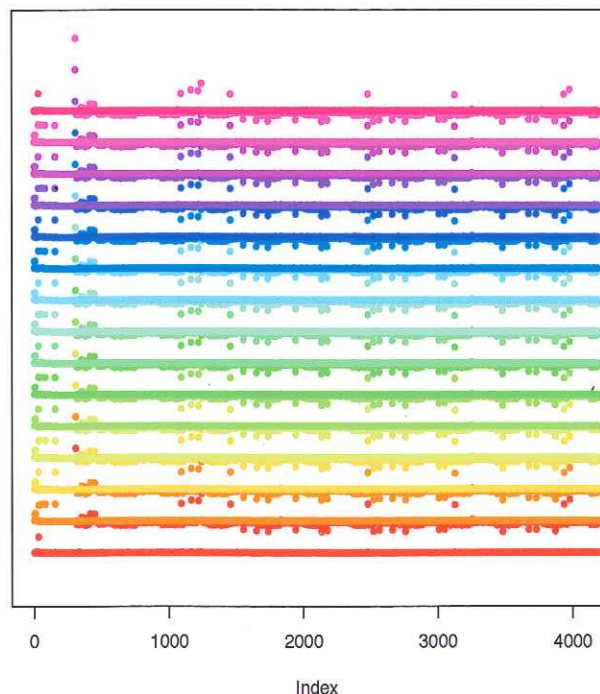
$$t_2: \begin{aligned} \text{if } 0 \leq t \leq 1 \quad g(t) &= \int_0^t 3(t-x)^2 dt \\ \text{if } 1 \leq t \leq 2 \quad g(t) &= \int_{t-1}^1 3(t-x)^2 dt \end{aligned}$$

$$t_3: \begin{aligned} \text{if } 0 \leq t \leq 1 \quad g(t) &= \int_0^t (t^3)^3 dt \\ \text{if } 1 \leq t \leq 2 \quad g(t) &= \int_{t-1}^1 (-t^3 + 3t^2 - 3t + 2)^3 dt \end{aligned}$$

5 Trials → distribution of varying critical path lengths, critical path jumps between ranks



Noise Variation by Rank



## Accomplishments

- load balanced synthetic Jacobi benchmark
- PMPI noise measurement and noise injector
- Noise simulator and replay engine built in R

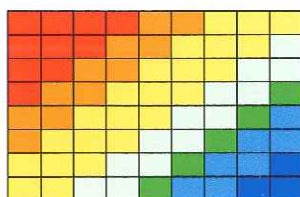
## Current Status

- Validating simulator with replay engine

## Goal

- Allow simulation of noise for proposed exascale architectures

```
main<-function(trials,procs,timesteps)
  for (all trials)
    generate histogram based on data observed on real machine
    return critical path
CP<-function(procs,probability,samples)
  for (all time steps)
    generate a noise distribution based on the probabilities and samples from real machine
    start time is the sum of the previous start time and the largest of noise value
    critical path is the max of the distribution of start time
max<-function(vec)
  given that each rank talks two immediate neighbors, rotate noise distribution up one and down one
  return largest execution time of a rank
```



Jacobi heat transfer illustration  
symmetrically through a grid.

```
main()
  determine how many rows will be allocated to each rank
  do {
    iteration function()
    find largest delta value resulting from all ranks
  } while (delta value resulting from all ranks > delta inputted by user)
iteration function()
  for (all rows and columns in subset grid)
    calculate the average of the sum of the neighboring cells
  return largest delta value resulting from subset grid
```