Modeling the Linux Block I/O Layer

M. Edward (Ed) Borasky

http://linuxcapacityplanning.com

LinuxCon, September 23, 2009

Where Is This Stuff?

http://github.com/znmeb/LinuxCon2009/tree/master/Modeling_the_Linux_Block_I-O_Layer_For_Enterprise_Applications/

Some Thoughts About Data

- "In God We Trust – all others bring data."
- "I like my data the same way I like my vegetables – raw or lightly steamed."
- "Give me data or give me death!"

More Thoughts About Data

Dr. Neil J. Gunther, *Guerrilla Capacity Planning Manual*

- “Data comes from the Devil, only models come from God.”
- “If the measurements don’t support your PDQ model, change the measurements.”

“The Problem” as “Traditionally” Stated

- “I have tons / reams / gigabytes of performance data. How do I make sense of them?”
- “How can I visualize performance of my application / server / enterprise?”

Visualization, Like Beauty, is in the Eye of the Beholder

Performance visualization is an exercise in exploratory data analysis, made easier by the existence of analytical models.

or

"You can observe a lot just by watching." – Yogi Berra

Exploratory Data Analysis

- "Let the data speak for themselves"
- Mostly visual / graphic tools
- Many are compute-intensive
- Often used when the investigator does not know much about what the data represent or how the variables are related

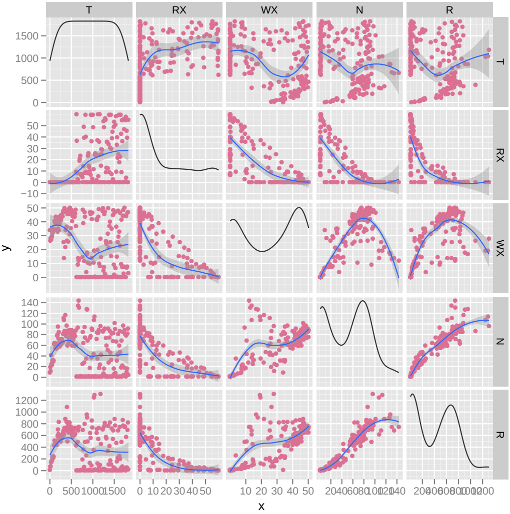
Exploratory Data Analysis Tools

- Box-and-whisker plots
- Scatterplot matrices, linked plots and brushing
- Kernel density estimation
- Kernel smoothing and quantile regression
- Classification (supervised learning)
- Clustering (unsupervised learning)
- Grand tours

Analytical Models

- Queuing networks, especially product form
- Stochastic Petri nets
- Process algebras

A Sample Scatterplot Matrix



What Are We Looking for in a Scatterplot Matrix?

- *Distributions* of the variables
  - Kernel density estimator on the diagonal
- *Clusters*
  - Groups of points in the scatterplots that represent distinct behavior patterns
- Linear or non-linear *relationships* between the variables
  - The shaded curves in the off-diagonal plots
  - Kernel smoothing / regression with standard error
  - In "traditional" EDA, we may not know these relationships. But for performance metrics, we often have workable models to guide us!

Capacity Function of a Load Dependent Queuing Center

$$C(N) = X(N)/X(1)$$

where

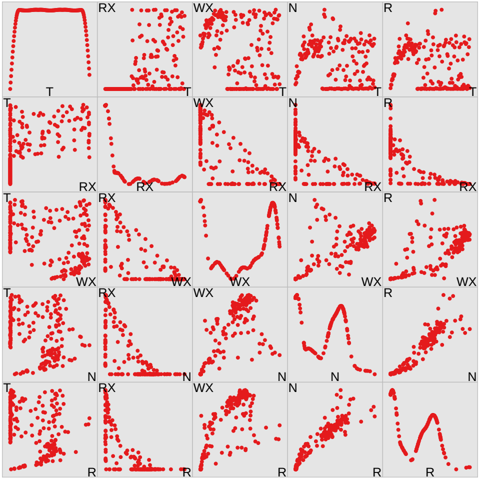
- N is the number of “customers”
- $X(N)$ is the throughput for N “customers”
- $C(N)$ is the capacity of the center for N “customers”

Note that the high-level model requires a numerical table of the capacity function!

In the Following ...

- Throughput is measured in megabytes per second
- "Customers" are I/O requests to the Linux block layer
- Data came from a complete *iozone* run using the deadline scheduler
- *iostat* samples were taken every ten seconds

Initial Scatterplot Matrix



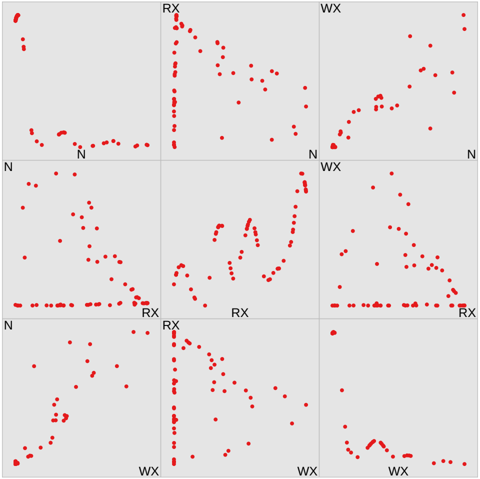
Notes

- Done with *ggobi* – linked plots and brushing are built-in
- Plots on diagonal are *average shifted histograms*
  - Can be linked with the other plots
- Variables are
  - T: seconds from beginning of benchmark
  - RX: read throughput in megabytes per second
  - WX: write throughput in megabytes per second
  - N: average queue length at the disk
  - R: average residence ("wait") time in milliseconds
  - Read capacity function is row 4, column 2
  - Write capacity function is row 4, column 3
- *Read and write capacity functions are very different!*

Getting the Read Capacity Function

- Write function is easier than read, so we focus on reads
- First, drop variables T and R
- Eliminate points where $RX < WX$
- Eliminate points where $RX = 0$

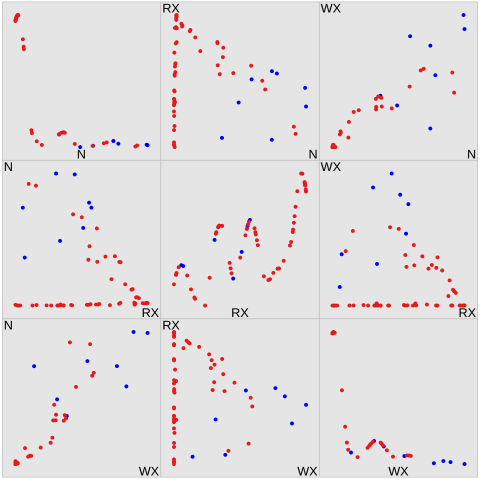
Reduced Scatterplot Matrix



Almost Done

- Function we want is row 1, column 2
- Rises to a sharp peak, then tapers off gradually
- Still has throughput, not capacity, on the Y axis
- Some “noise points” need to be removed
- So we remove the points we don’t want by *shadowing* them with the brush

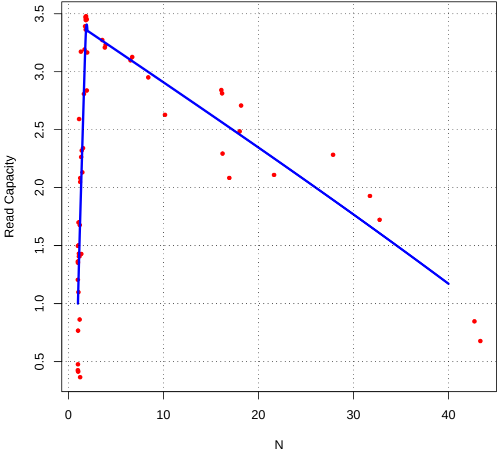
Scatterplot Matrix After Shadowing



Notes

- Shadowed points are in blue
- Now we will make an ordinary scatterplot and use kernel regression to fit the function
- Because it has a sharp peak, we will fit twice, once to the left and once to the right

Read Capacity Function



Write Capacity Function