Complexity

Mendel Rosenblum

CS110

Agenda for Today

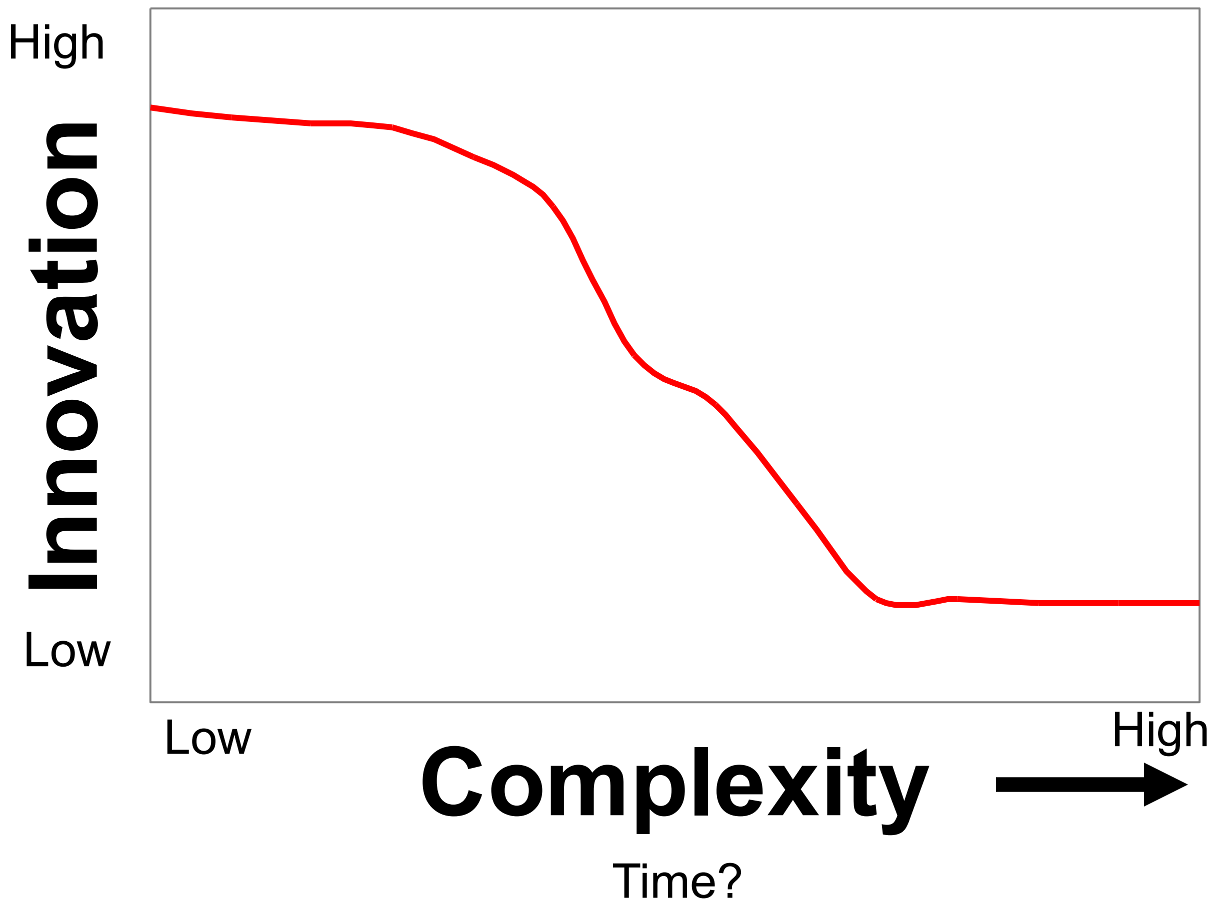
- Complexity
  - Why you should fear it
  - Problems with it
  - Signs of it
  - What can be done about it

Human understanding is limit

- Computer systems are only limited by the ability to of humans to understand.
  - Bounds imposed by physical laws and theory are huge in comparison
- Others man-made systems have noise
  - Digital systems don't

Complexity

Impact of Complexity



Common problems in Complex Systems

Problem classification categories:

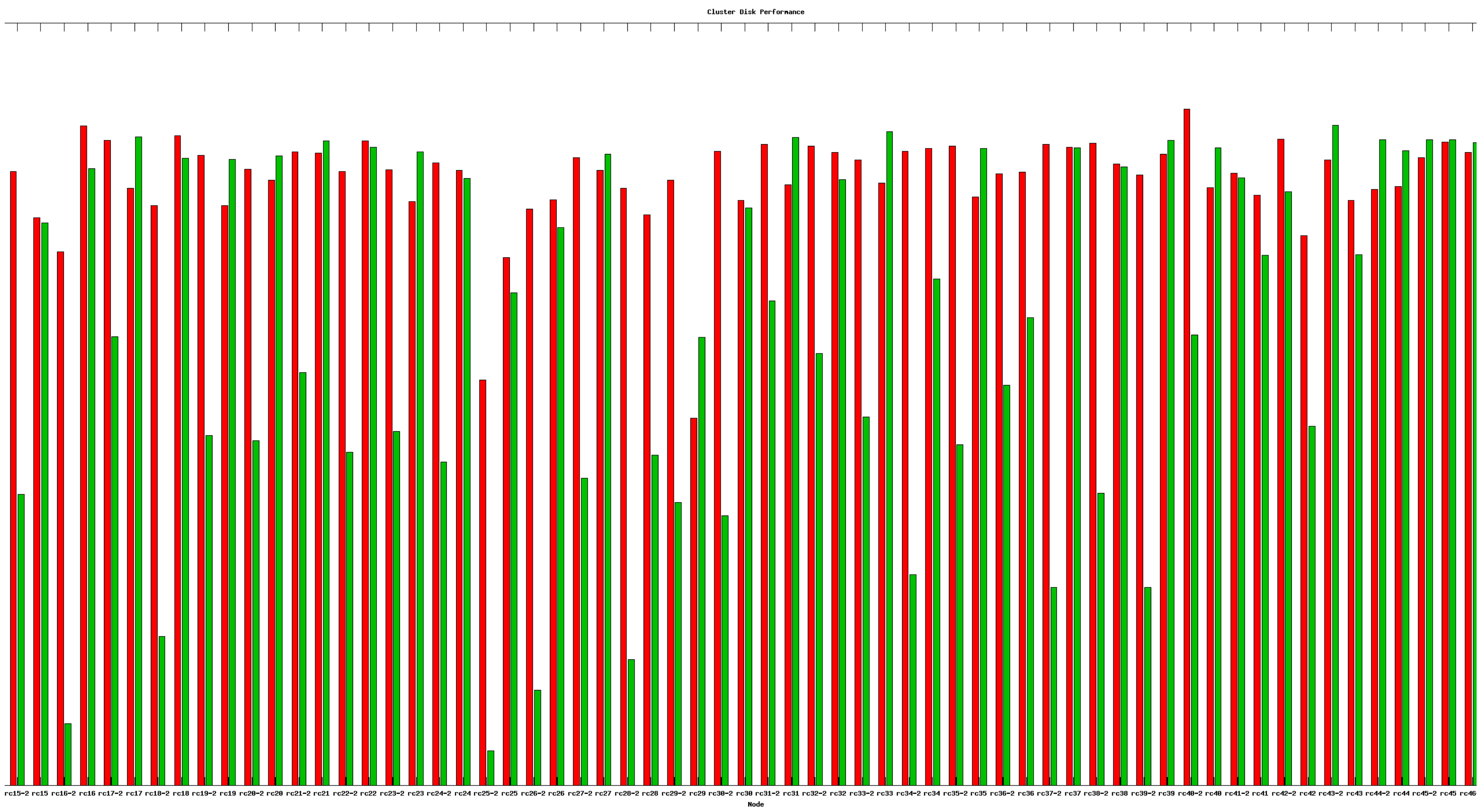
- Emergent properties
- Propagation effects
- Incommensurate scaling
- Trade-offs

Emergent properties

- Properties not evident from components
  - Surprises, frequently we call them bugs
    - Security, reliability, and performance
  - Term from social sciences

    What was that jury/mob/group thinking?
- Examples from computer systems:
  - Shaking disk drives

Disk Bandwidth



Propagation effects

- Change or problems in one place effects some other part of the system
  - Frequently chaotic: Small changes, big effect
- Root cause analysis
- Examples:
  - AMD’s loop instruction optimization
  - How to ruin someone's day with a small change

Folk wisdom:

“There are no small changes in large systems”

“A distributed system is one in which a machine that I never heard of prevents me from getting work done”

Incommensurate scaling

- As a system increases in size or speed, not all parts follow - something breaks
  - Something becomes the bottleneck
- Example:
  - CPUs faster, memory bigger, disk not faster
- Factor of ten increase frequently breaks things

Trade-offs

- Limited amount of a currency
    - Examples: money, energy, developer time, etc.
  - Maximize it
  - Spent it wisely

  Implies decisions: Trade-offs
- In computer systems:
  - Performance optimizations
  - *Waterbed effect*: Push here, pops up somewhere else
    - "Can I get away with …"

Trade-offs are what system designers do

No simple metric of complexity

- How do you evaluate complexity of a software system?
  - Complexity theory not helpful (e.g. big-O)
  - KLOC (Kilo Lines Of Code)
  - Cyclomatic complexity
    Independent paths through program
- No explicit cutoff – Too complex to build
  - Systems labeled too complex have worked
  - Systems seemly achievable failed

Signs of Complexity

No definitive litmus test. Some signs:

1. Large number of components
   – Can you fit the entire thing in your head?
2. A team of designers, implementers, or maintainers
   – Need multiple people to get their heads around it.
3. Large number of interconnections
   – Interactions between things get you
4. Many irregularities
   – Special cases get you
5. A long description
   – Shortest complete description is long

Is the following program complex?

int main(void) { printf("Hello World\n"); }

  – One line of code, 82 bytes of code
  – No, can fit in my head.

- But:
  – printf calls the write system call
    - Millions of lines of code in OS.
  – Runs on an x86 processor:
    - Billion of transistors on the chip.
  – Transistors are made of silicon
    - Many electrons, protons, …
  – Sub-atomic particles are …
- Answer: **Abstraction**

Main Sources of Complexity

1. Cascading and interacting requirements
   - Your company's product marketing at work
     - Each feature add complexity
     - Interactions between features add complexity
   - New features require change, change adds complexity
     - Backward compatibility
     - Bug fix introduces more bugs
   - Look at the QA department of any software vendor
2. Maintaining high utilization
   - Example: Multiple users on same computer
   - Tricky optimizations

Principle of escalating complexity

Adding a requirement increases complexity out of proportion

- Adding features increase the overall system complexity more than the complexity of the feature
  - Examples:
    - US tax code
    - Most popular packaged software products

Generality vs Specialization

- A system specialized for a particular task is frequently better than using a more general tool
  - Eg. Powerpoint vs Word for a presentation
- A too specialized system may have too small of user community
  - Tend is to expand functionality to get more users
- Fundamental tradeoff facing system designers
  - How rich is the interface to the subsystem you are designing?

Principle: Avoid excessive generality

If it is good for everything, it is good for nothing

- Trying to do everything for everybody ends up doing everything poorly

Examples:

A vehicle that travels on road, water, and sky.

Some software products…

Principle: The law of diminishing returns

The more one improves some measure of goodness, the more effort the next improvement will require

- Taken from Economics

- Examples:
  - Performance optimizations
    - You get the low hanging fruit, the rest take more work
    - Trickier, more complex algorithms
  - Complex control needed for high utilization
    - Eg. Distributed file system performance

Sample Exam Questions

- A programming style that demands frequent use of assert() statements is an attempt to address which problem?

- Does the statement that **propagation effects** can cause **emergent properties** in systems make any sense?