Experimental Approaches in Computer Science

Dror Feitelson
Hebrew University

Lecture 1 -- Introduction

The complexity of sorting is *O(n log n).*

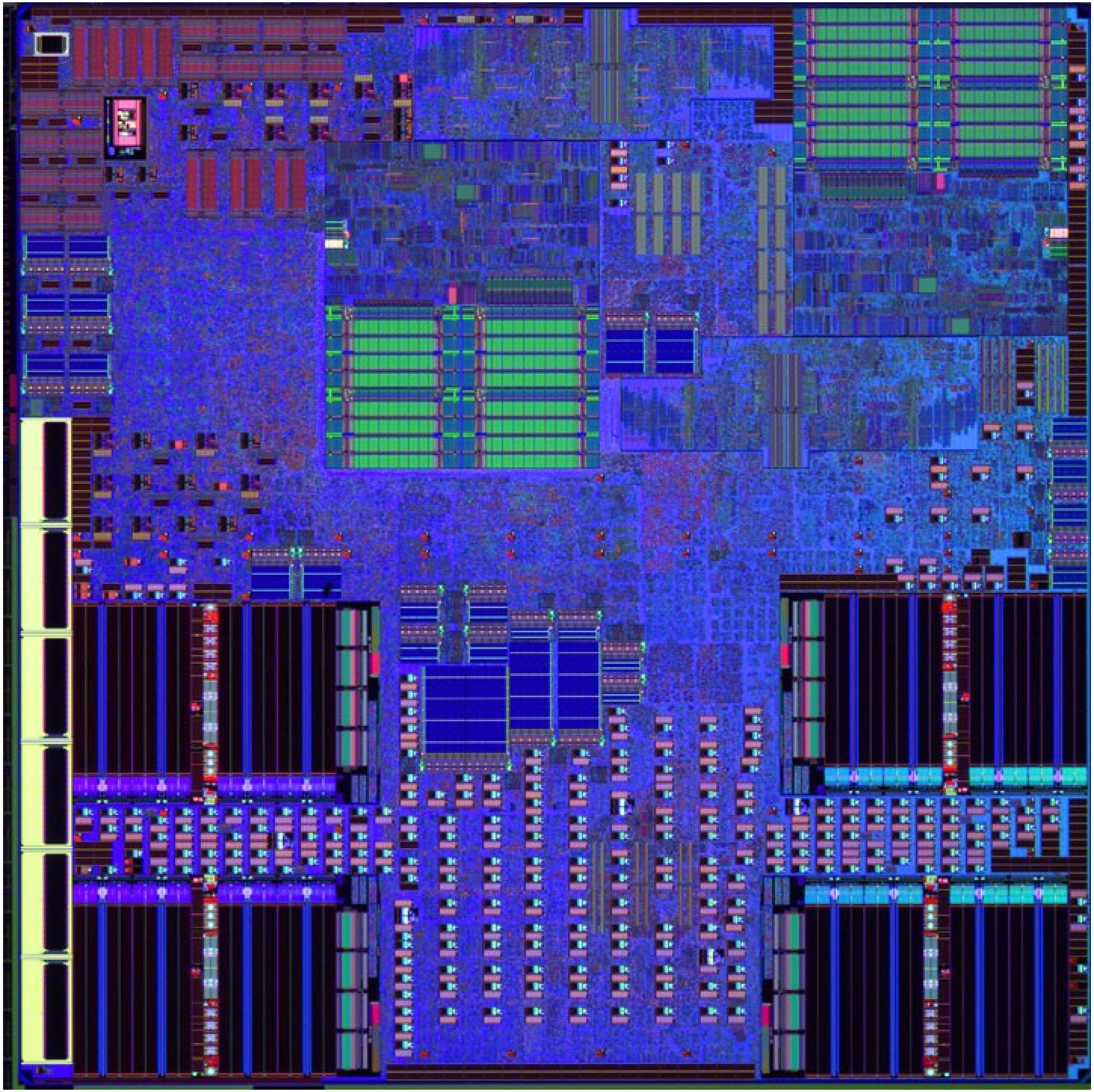
Have you seen anyone actually measure it for different values of *n* to verify that the relationship indeed holds?

(you will in one of the exercises...)

Computer science is based on theory in the context of abstract models which are assumed to reflect reality

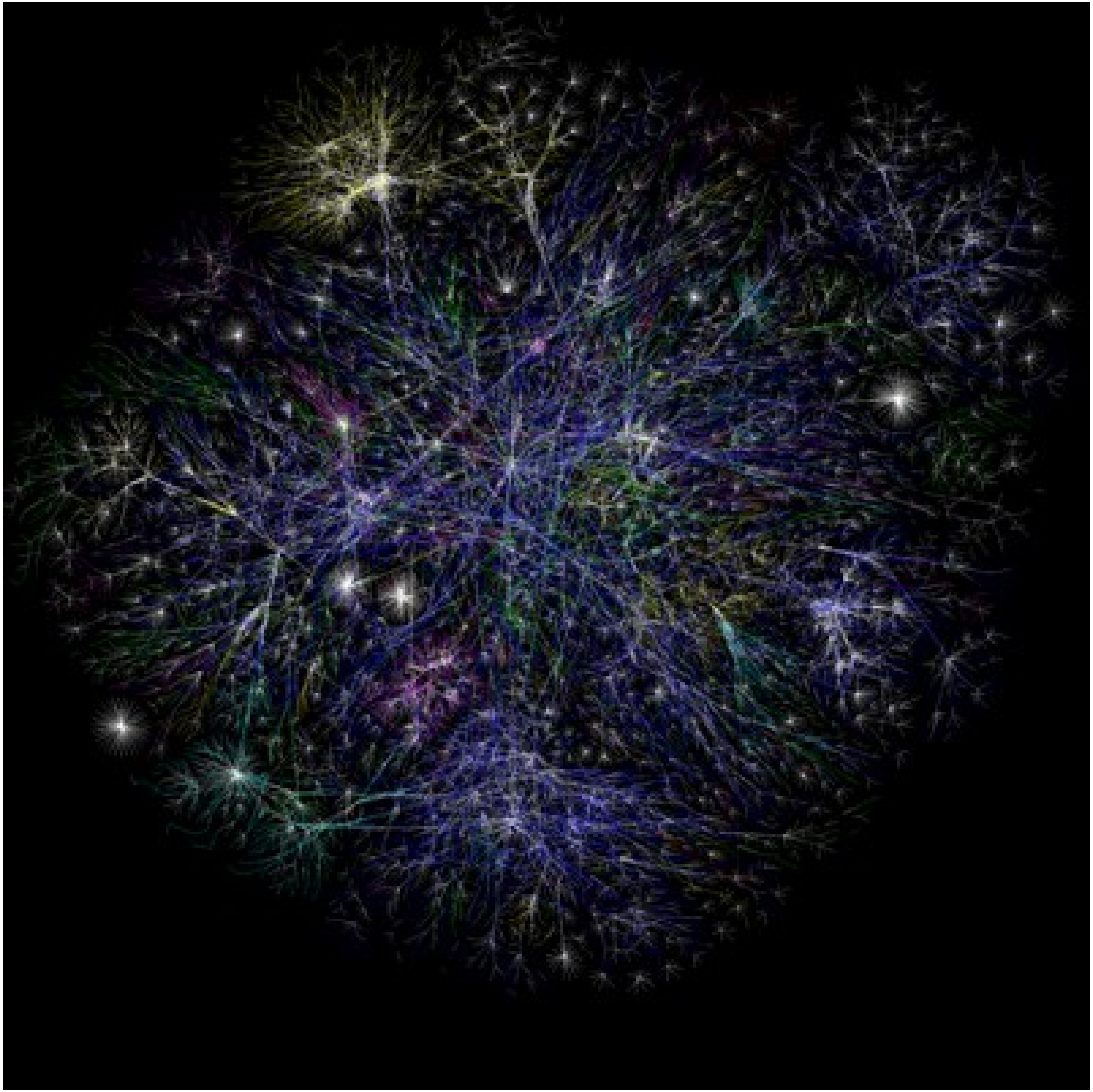
The justification: computers are not natural phenomena, but are designed and built by humans, so we know how they work.

Do you believe this for modern micro-processors?



Source: IBM BlueGene/L

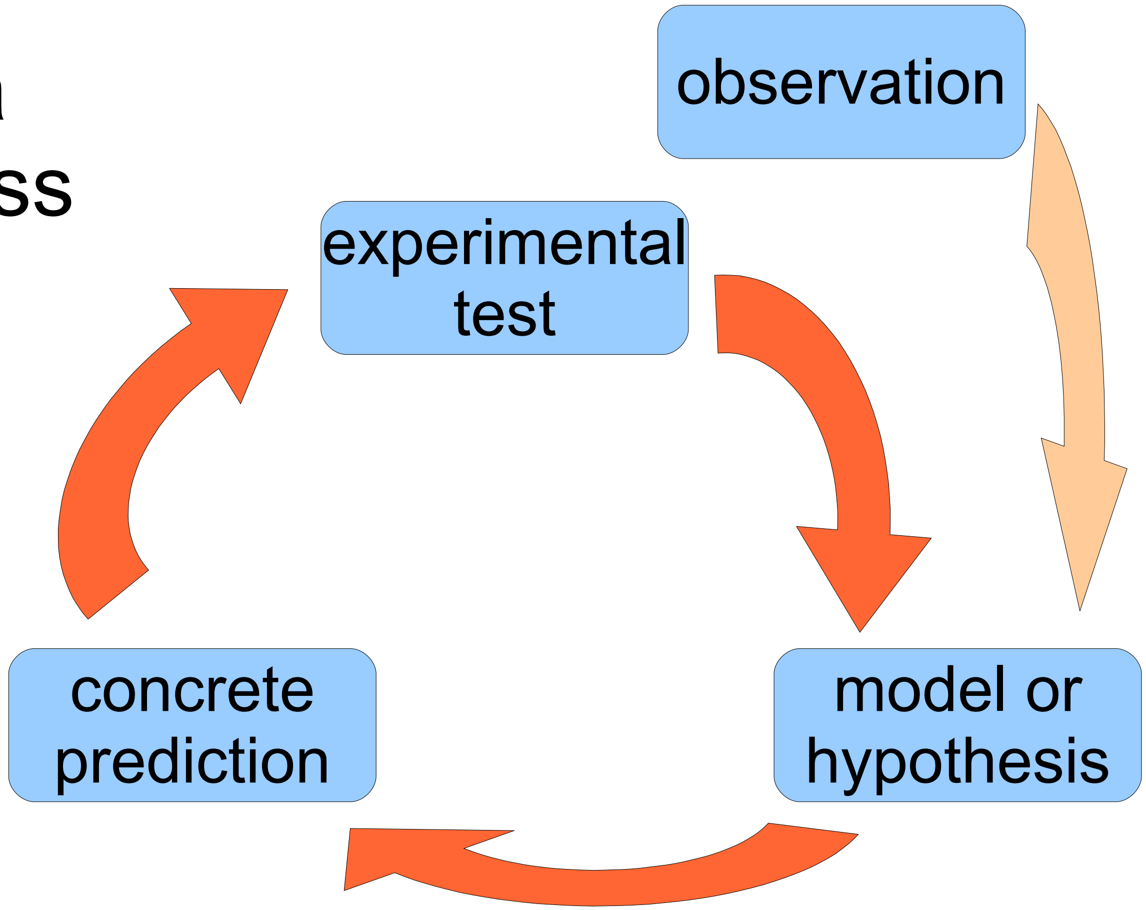
Or the Internet?



Source: www.opte.org

"Real" science is based on observations
which lead to models and theories
which enable hypotheses and predictions
which can be verified experimentally

And this is a
cyclic process



Two important points:

Theories can be refuted by experiments.
This distinguishes science from religion.

"A theory which cannot be mortally endangered cannot be alive"

W. A. H. Rushton

Experiments can be reproduced by others,
in order to verify the results.

Claim:
all this is relevant to
computer science

Experimentation in computer science:

- Know the world in which we live
- Complement theory
- Support design and engineering

Know the world example 1: Locality

We all know that computer programs display locality. But,

- Given two programs, how do you know which has more locality? How do you quantify locality?

Actually, there are a number of ways.

- Stack distance
  - Maintain all previous references in a stack
  - Upon each access, note it's depth in the stack
  - Strong locality implies references will be found near the top
- Autocorrelation function
  - In particular, is each reference correlated with the next reference?
- Number of combinations observed
  - Are all possible combinations of successive references actually observed?

Know the world example 1: Locality

We all know that computer programs display locality. But,

- Given two programs, how do you know which has more locality? How do you quantify locality?
- What is the relationship between different metrics of locality?
- How much locality is needed to make caching effective?

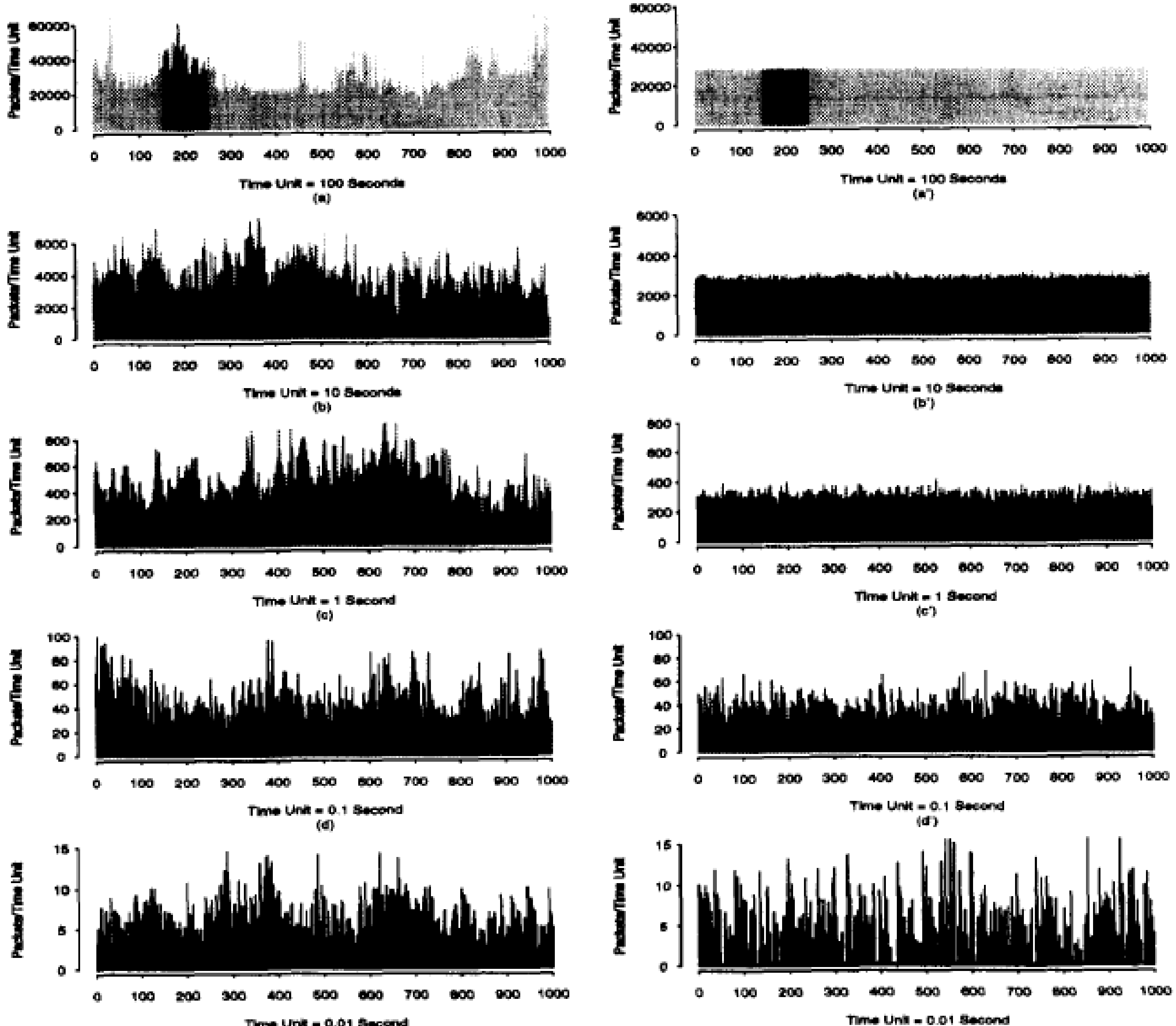
Know the world example 2: Self-similar traffic

For many years network traffic was assumed to be Poisson

- This means that arrivals occur uniformly and at random

Turns out that this is not the case: network traffic is self-similar

- It is bursty at many different time scales
- It does not average out
- There are long-range correlations
- Important for provisioning buffers and QoS



From Leland et.al, On the self-similar nature of Ethernet traffic, *IEEE/ACM T. Networking*, 1994

Complement theory example: thresholds in NP-complete problems

Given a Boolean formula in conjunctive normal form on *n* variable with *m* clauses and *k* variables per clause, the *K-SAT* problem asks whether it can be satisfied.

This can be done in polynomial time for $k=1,2$

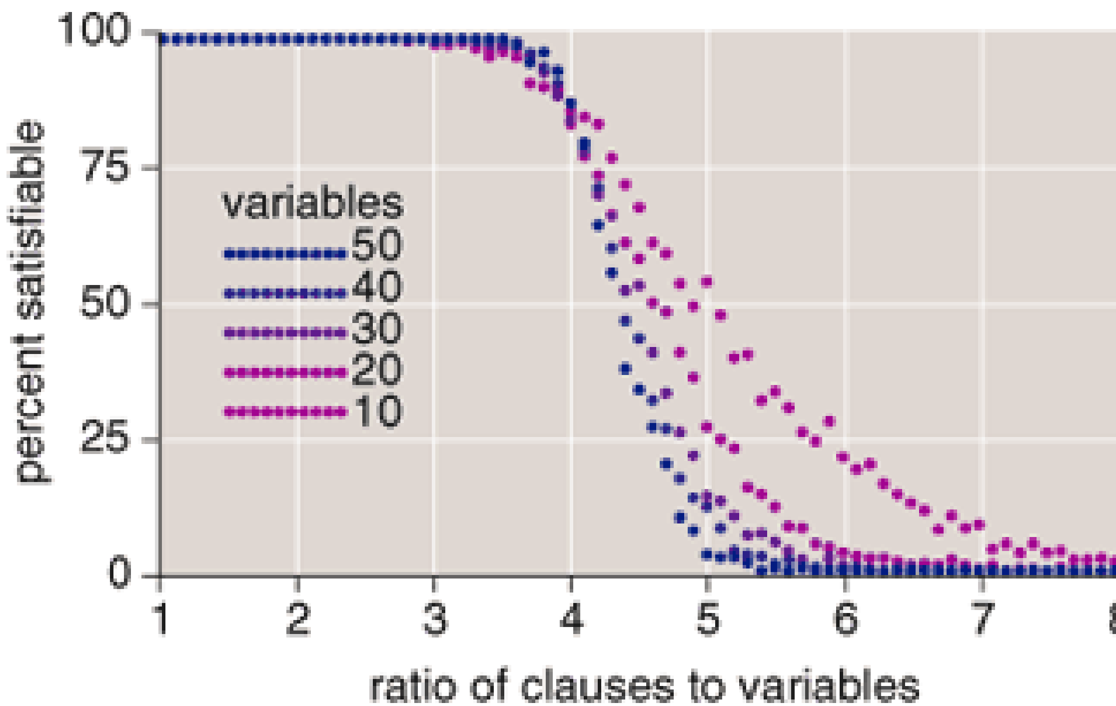
It is NP-complete for $k \geq 3$

However, some instances with $k \geq 3$ turn out to be very easy.

Given a formula generated at random, you can apply the following heuristics:

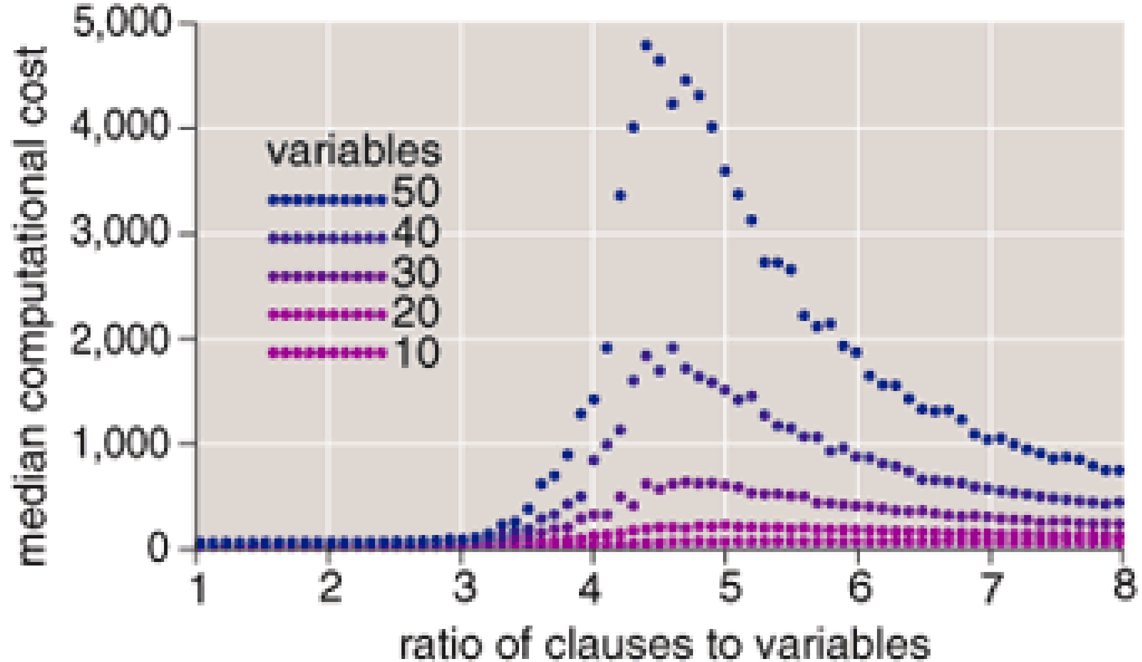
- Try to find a satisfying heuristic by setting variables arbitrarily and using backtracking
- Try to find a simple proof that the formula is not satisfiable, e.g. if it contains conflicting clauses

It turns out that the probability that the formula is satisfiable depends on *m/n*, and displays a strong threshold effect.



From Hayes, Can't get no satisfaction, *American Scientist*, 1997

Moreover, the difficulty of deciding if a formula is satisfiable depends on the distance from the threshold:

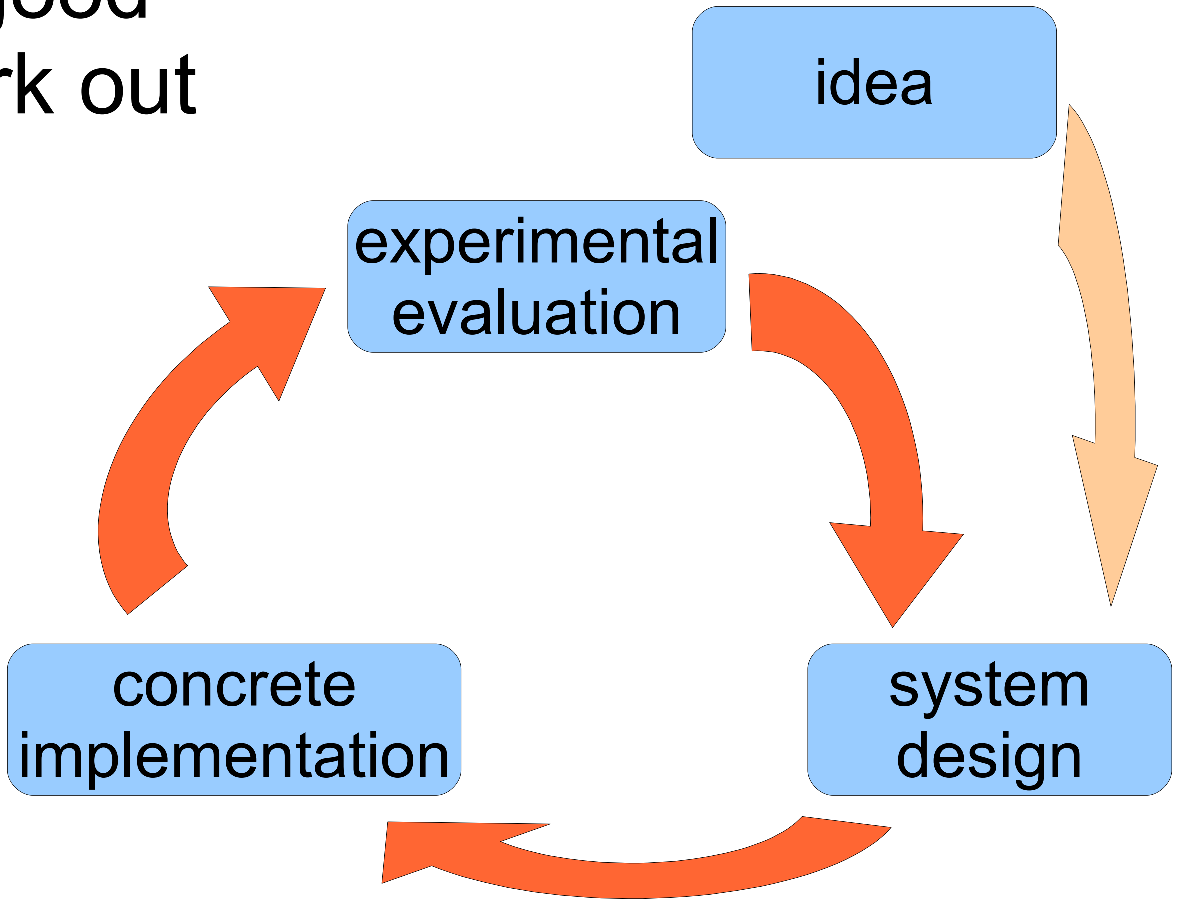


So now we have a finer classification: by characterizing how the threshold depends on k, we can say which problems are easy or not.

Support design and engineering

Ideas that look good don't always work out in practice

They need to be tested in realistic conditions

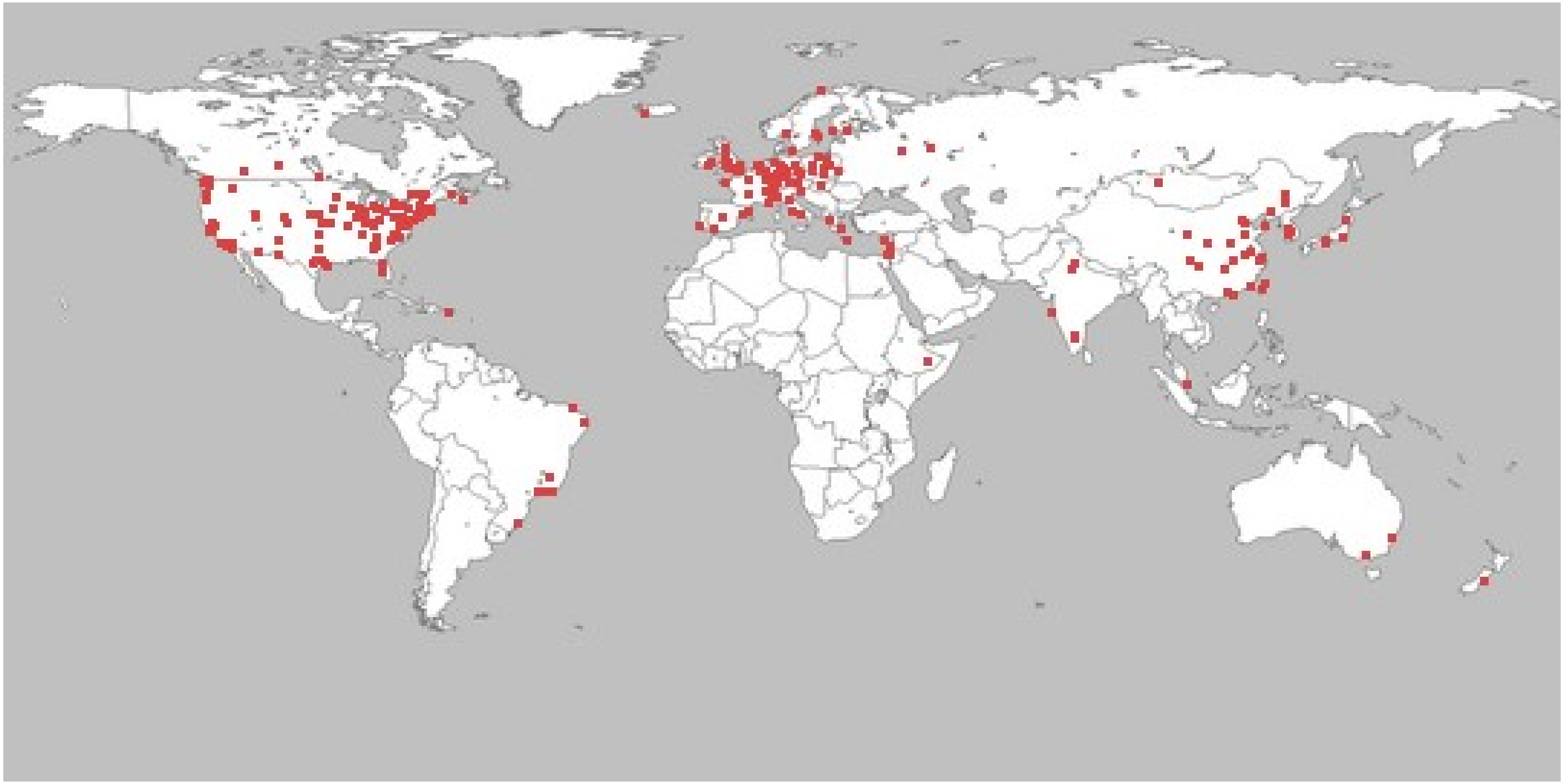


Experimental engineering example: PlanetLab

Assume you have a great new idea for an Internet service or protocol. How can you test it?

- Need large scale with many nodes
- Need realistically high latencies
- Need interaction with other types of traffic
- Need feedback from actual usage
- Need not to disrupt existing applications

Answer: use the PlanetLab overlay network infrastructure.
As of January 2007, PlanetLab consists of 753 nodes at 363 sites (including here).



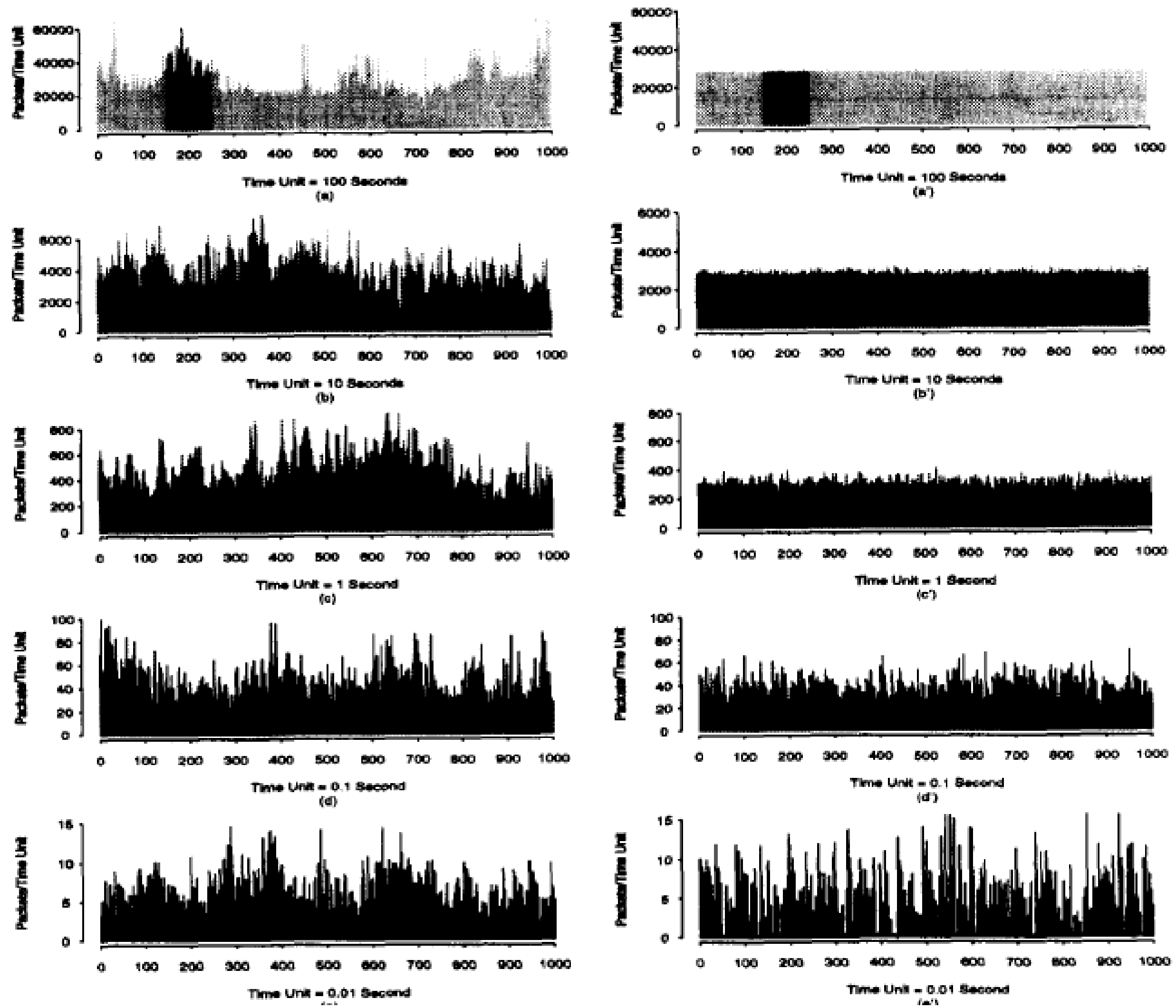
From www.planet-lab.org

Experimental activities:

- Observations and measurements
- Forming hypotheses ans testing them
- Reproducing results

Observation and measurement

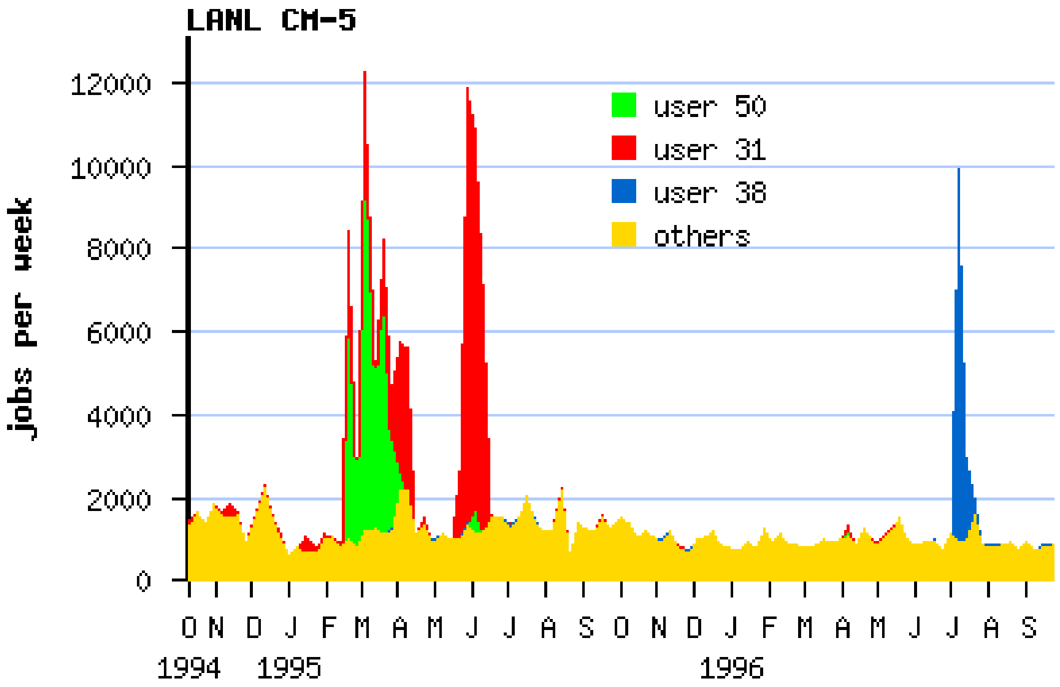
- Collect data
  - For example, trace all the packets in a local network

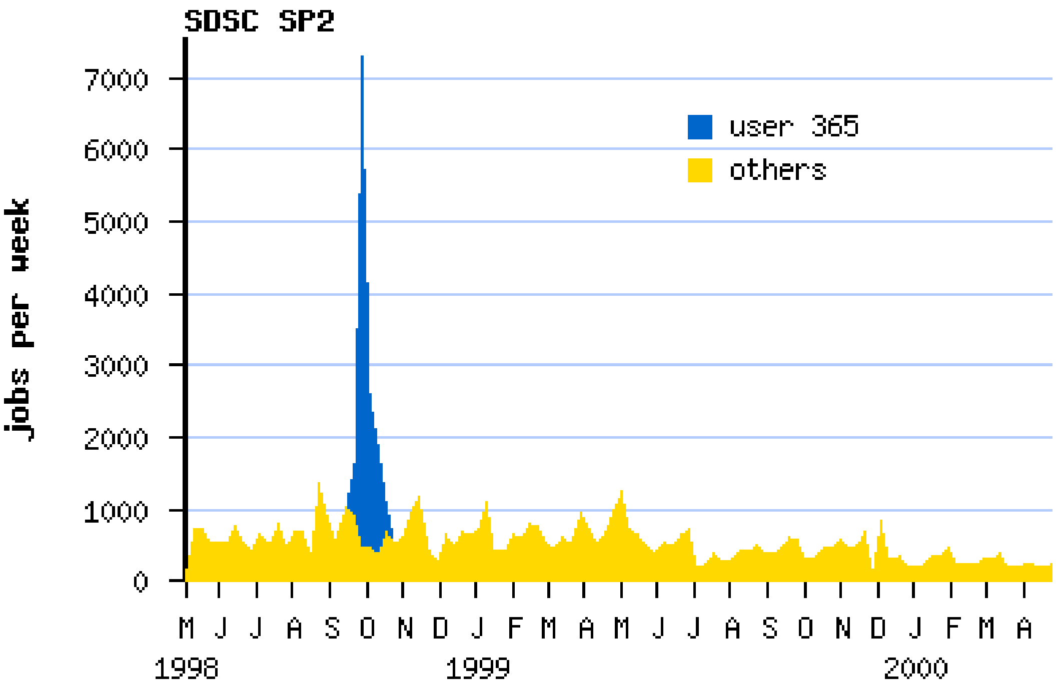


Observation and measurement

- Collect data
  - For example, trace all the packets in a local network
- Clean the data from outliers and "bad" data

Example: huge flurries of activity by single users on parallel supercomputers





Observation and measurement

- Collect data
  - For example, trace all the packets in a local network
- Clean the data from outliers and "bad" data
  - For example, flurries
- Perform point measurements

Small issue of what to measure and how

- Need to define appropriate metrics
  - Time and throughput are easy
  - Locality needs some thought
  - And how do you measure the degree to which virtual machines are isolated from each other?
- Need to perform measurements reliably
  - Avoid interference
  - Take all effects into account
- Need to record exact conditions used

Observation and measurement

- Collect data
  - For example, trace all the packets in a local network
- Clean the data from outliers and "bad" data
  - For example, flurries
- Perform point measurements
- Experiments with humans
  - They both build and use computer systems
- Display results clearly in graphs
- Share data

Hypotheses (NOT)

- The most famous hypotheses are actually "meta-theories"
  - The theory of evolution
  - The conjecture that P≠NP
  - The claim that intelligence can be achieved by extensive search
- These are summaries that fit a large body of experience
- This is not what we are talking about

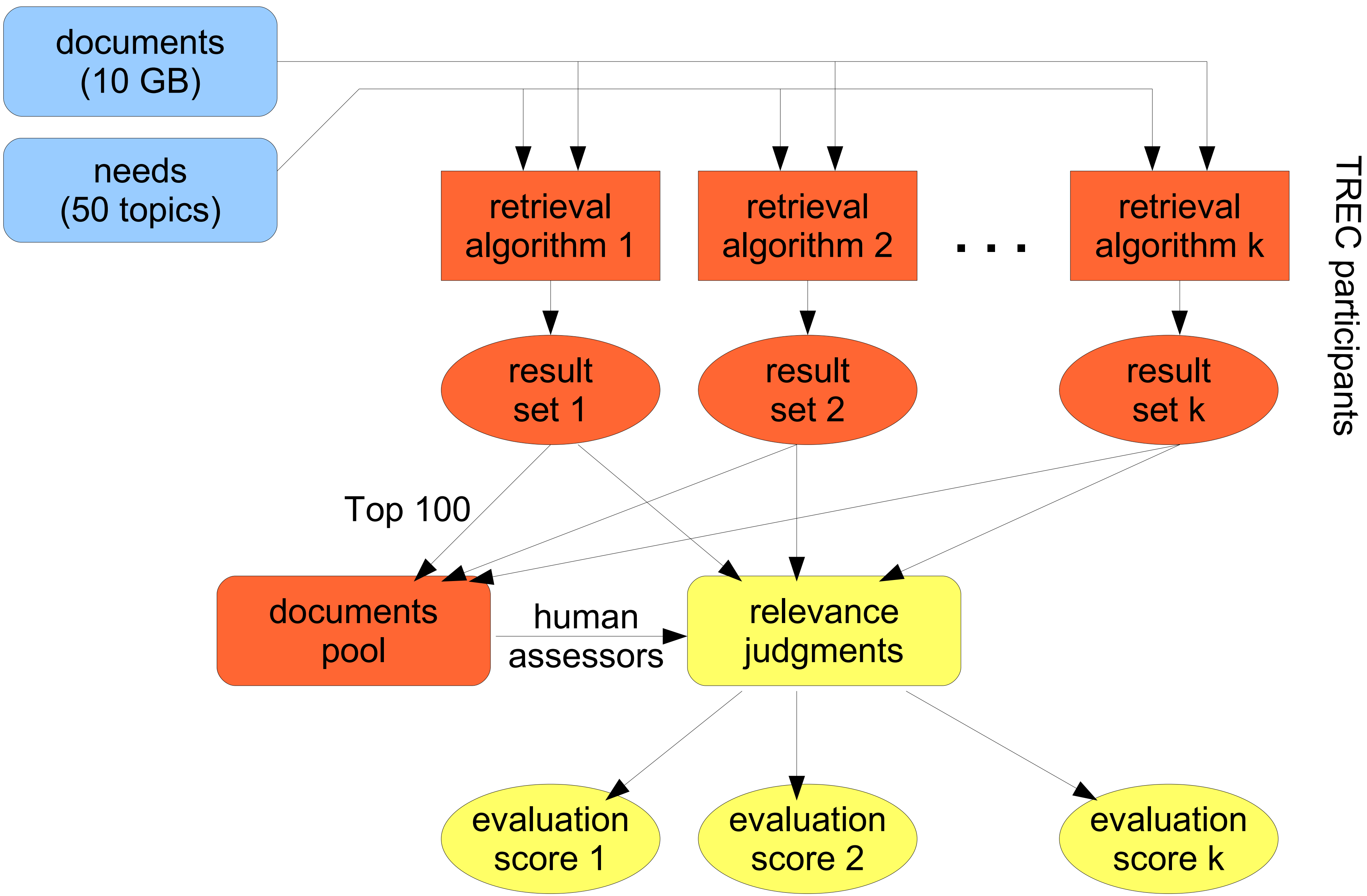
Hypotheses

- A model that tries to explain observations
- And enable predictions
- Metric for good hypotheses: can be tested experimentally
- In particular, can be refuted
- This enables the most rapid and consistent progress

Reproducibility

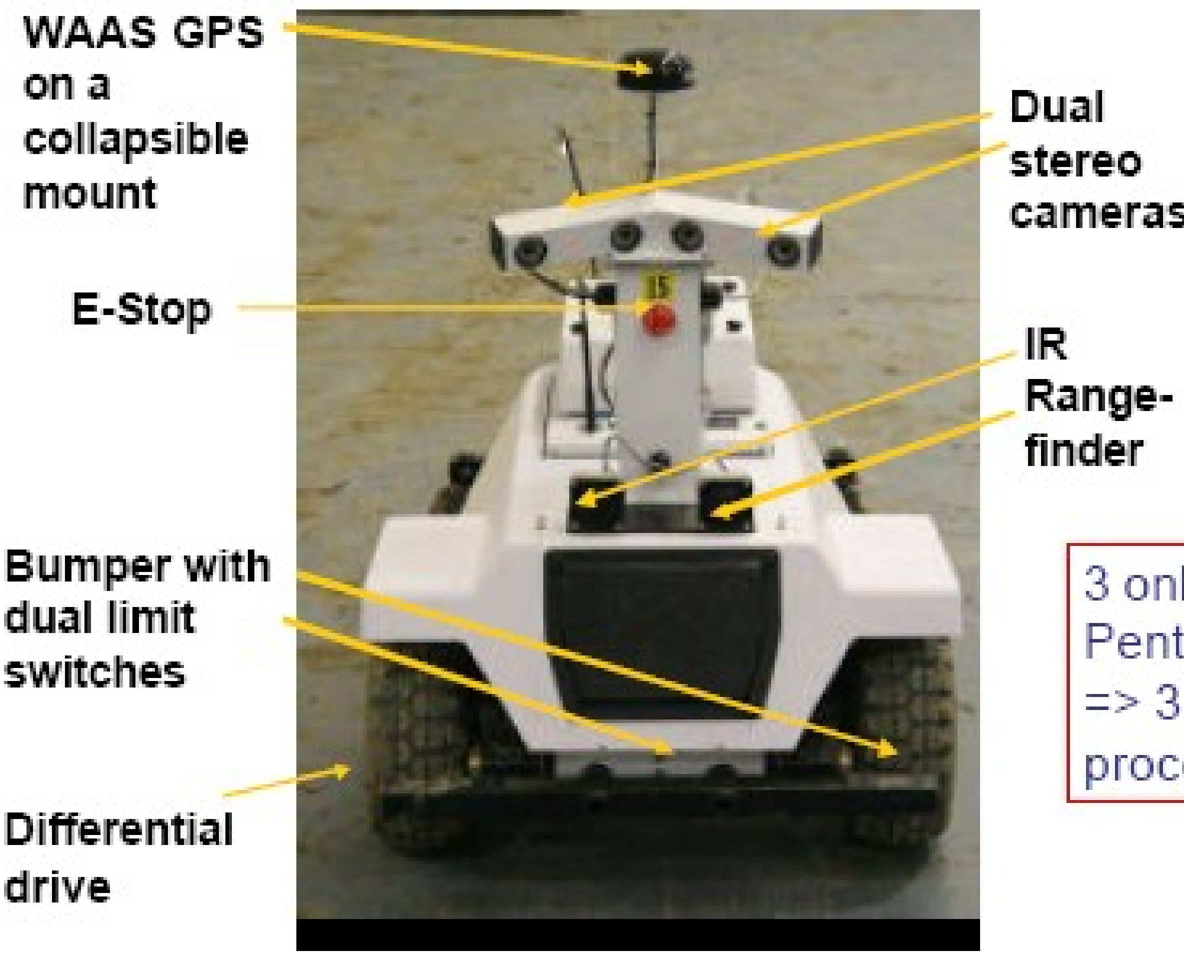
- Verify that published results are correct
  - This is the least important aspect
- Identify exactly what conditions need to be reproduced to get the same results
  - Improves our understanding of cause and effect
- Foster progress by a concentrated effort of multiple teams
  - TREC
  - DARPA robotics program

Structure of a TREC track



Based on Voorhees, TREC, *Comm. ACM* 2007

DARPA learning robots program: each team gets identical platform, can focus on robot software rather than on platform development, ships software for testing at DARPA



3 onboard mobile Pentium computers => 3 dual core processors

From Jackel et al., Structuring DARPA's robotics programs, *Comm. ACM* 2007