computer systems performance
lecture 1
intro & experimental design

Pınar Tözün

February 4, 2025

Some experimental design slides are influenced by Philippe Bonnet's slides on the same topic
2^k experiment slides are influenced by Zsolt István's slides on the same topic

agenda

lecture

- course overview & logistics
- performance analysis
- experimental design

exercises

- overview of first project
- group selection
- access to course server
- some useful tools for running experiments on remote servers

data systems specialization

DASYA
Data-Intensive Systems and Applications

spring

Computer Systems Performance
- experimental design
- layers of computer systems & their performance impact

Internet of Things
- sensor networks
- edge devices
- embedded systems

fall

Readings in Data Systems
- paper reading & discussion
- guest lectures

How to Build Data Systems
- hands-on

Big Data Management
- scalability of analyzing data

dasya.itu.dk

all 7.5ECTS

who are we?

lecturer:
Pınar Tözün – pito@itu.dk @4E06

TA:
Emil Houlborg – ehou@itu.dk

my scientific intro

IT UNIVERSITY OF COPENHAGEN — associate professor

Copenhagen, Denmark
2018 – present

IBM — research staff member

San Jose, CA, USA
2015 – 2018

EPFL — PhD student

Lausanne, Switzerland
2009 – 2014

KOÇ UNIVERSITY — BSc student

İstanbul, Turkey
2005 – 2009

who are we?

lecturer:
Pınar Tözün – pito@itu.dk @4E06

TA:
Emil Houlborg – ehou@itu.dk

- last semester MSc student @ITU Computer Science
- specialization: data systems
- thesis: flexible data placement for database systems on modern SSDs

who else? – guest lectures

internal

Ties Robroek
postdoc @ITU

Ehsan Yousefzadeh-Asl Miandoab
postdoc @ITU

Robert Bayer
PhD @ITU

external

Bjørn Kisbye Engsig
Performance Architect
@Oracle

Tilmann Rabl
Professor
@HPI, Germany

Simon Lund
Software Engineer
@Samsung Research Denmark

other course trivia

reading:
book chapters & research papers
links/pdfs will be posted at LearnIT for each week

project assignments: to be done in up to 3 people groups
#1: will be available later today
progress to be presented on March 4 (for feedback)
#2: starts around March 11
will give you options to pick from,
you can also come up with something of your own
progress to be presented on April 29 (for feedback)

other course trivia

exercises:
hands-on tasks to do
help with the mini-projects & required tools
presentations for the assignments to get feedback

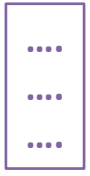
exam:
oral exam with hand-in on June 16-17
hand-in submission deadline is on May 23
hand-in = reports & code for the two projects

tentative course plan

Date	Week#	Lecture topic
Jan 28		**Break**
Feb 4	1	Intro, Performance Analysis, Experimental Design
Feb 11	2	Queuing Theory, Common Mistakes, Plotting Graphs
Feb 18	3	Hardware: Parallelism
Feb 25	4	Hardware: Memory Hierarchy
Mar 4	5	Operating Systems Overview
Mar 11	6	Hardware: Acceleration
Mar 18	7	Guest Lecture: Tilmann Rabl
Mar 25	8	Guest Lecture: Bjørn Kisbye Engsig
Apr 1		**Break**
Apr 8	9	Hardware: Persistent Storage
Apr 15		**Easter Break**
Apr 22	10	Tiny Benchmarking
Apr 29	11	Data Centers & HPC
May 6	12	Surge of AI and performance challenges
May 23		Exam Hand-In Deadline
Jun 16-17		Exam

systems stack overview

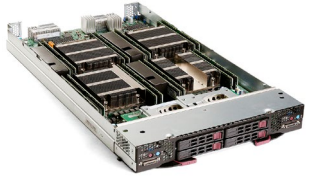
mapping to course contents



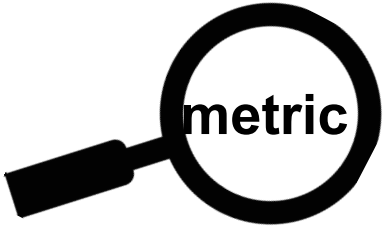
application — e.g., online shopping page, database system, code to read/write a file, etc.

operating system — e.g., linux, windows, etc.

hardware — e.g., intel server, disks, etc.



investigating systems performance



workload

system
(k parameters,
l levels)

methodology to design experiments

tools to run experiments & collect results

applications

operating system

hardware

understanding systems layers to interpret results

agenda

lecture

- course overview & logistics
- performance analysis
- experimental design

performance analysis

what is it?

in this course:

analysis of computer systems with respect to certain metrics to serve a scientific goal

performance analysis

why is it useful?

- identifying performance requirements
- comparing different systems / methods / design alternatives
- system tuning – optimal values for different parameters
- bottleneck identification
- characterizing workloads / applications
- capacity planning
- forecasting for future workloads
- …

performance analysis

how is it done?

in practice, a continuous process with all these steps!

- *analytical modeling*
  - analyze things with pen and paper
  - easy and low cost
  - but not very accurate in practice (models require you to simplify stuff)
- *simulation*
  - when measurements with the real thing aren't possible or very costly (e.g., trying out new hardware idea)
  - higher accuracy than analytical modeling
  - can take long time & not as accurate as a well-done measurement
- *measurement*
  - experiments with the real thing → the most convincing results
  - but need to be careful to not to end up with misleading results

agenda

lecture

- course overview & logistics
- performance analysis
- experimental design

designing experiments …

… or experimental methodology

systems

workloads

metrics

experiments

designing experiments

2 hardware systems being compared

Table 1: Server parameters

Processor	Intel(R) Xeon(R)	ARM Cortex-A57
#Sockets	2 (only one socket is active)	1
#Cores per Socket	8	8
Issue width	4	4
Clock Speed	2.00GHz	2.00GHz
Main memory	256GB	16GB
L1I / L1D	32KB / 32KB (per core) 16-cycle miss latency	32KB / 32KB (per core) 15-cycle miss latency
L2	256KB (per core) 40-cycle miss latency	256KB (per pair of cores) 80-cycle miss latency
LLC	20MB (shared) 170-cycle miss latency	8MB (shared) 175-cycle miss latency

parameters

collection of hardware, software, & firmware

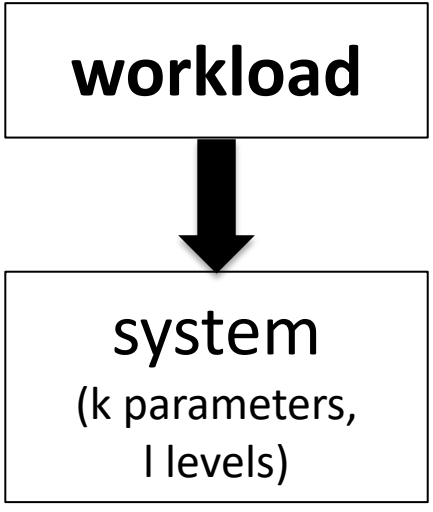
system
(k parameters, l levels)

examples for levels:
#cores: 1, 2, 4, 6, 8
memory to use: 1GB, 10GB, 100GB

figure from: Sirin et al. "OLTP on server-grade ARM"

designing experiments

requests made by the users of the system



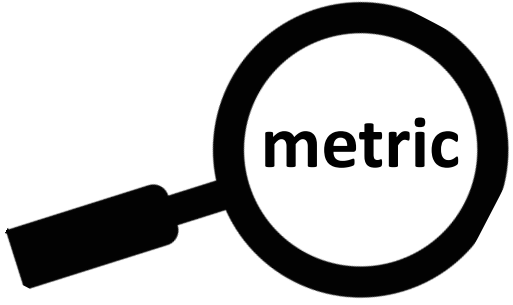
if system = hardware
workload could be operating system, database system, etc.

if system = deep learning framework
workload could be sentiment analysis, imagine classification, etc.

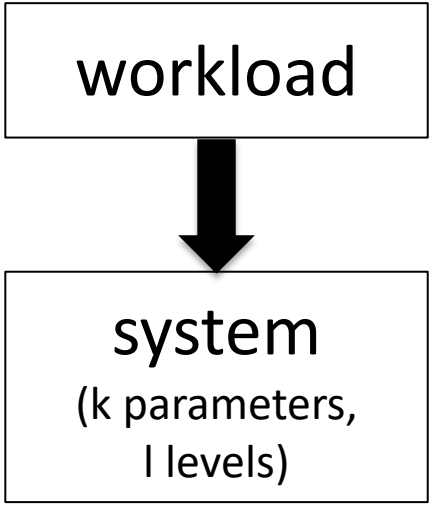
workload options:

- synthetic (e.g., benchmarks)
- trace-based (especially in simulation)
- actual run (e.g., real-life application)

designing experiments



criteria used to evaluate the performance of the system



most common

- **throughput**

requests processed in unit time

- **latency**

time between stimulation and response

- **energy**

watts being spent on a request

- **memory footprint**

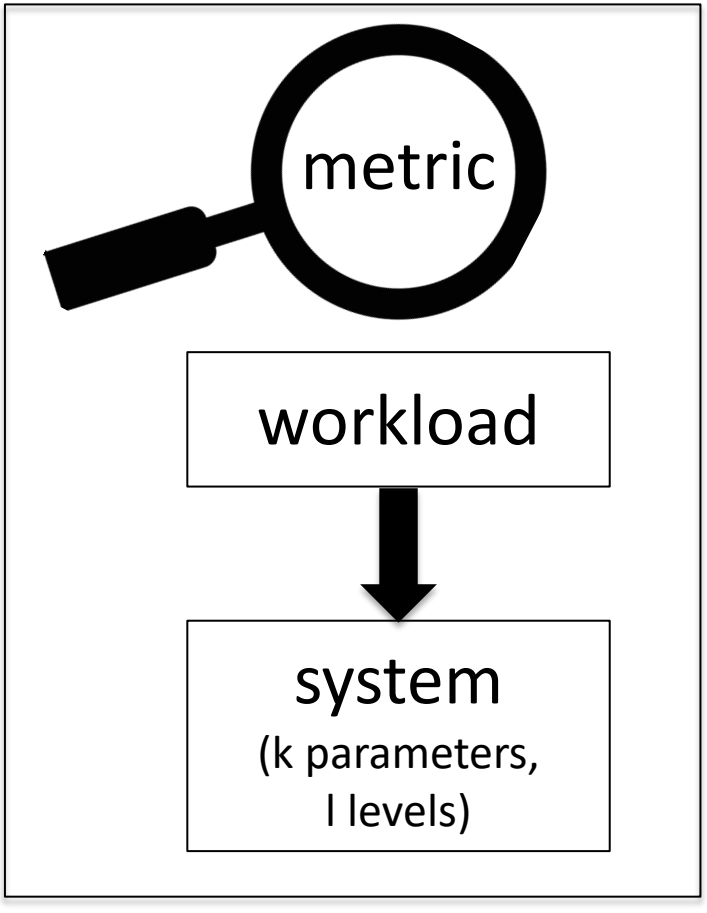
memory space being used

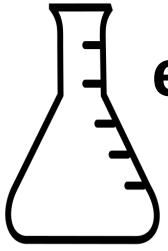
- **CPU utilization**

% of cores being used

- **ease of use**, **cache misses**, **security**, …

designing experiments





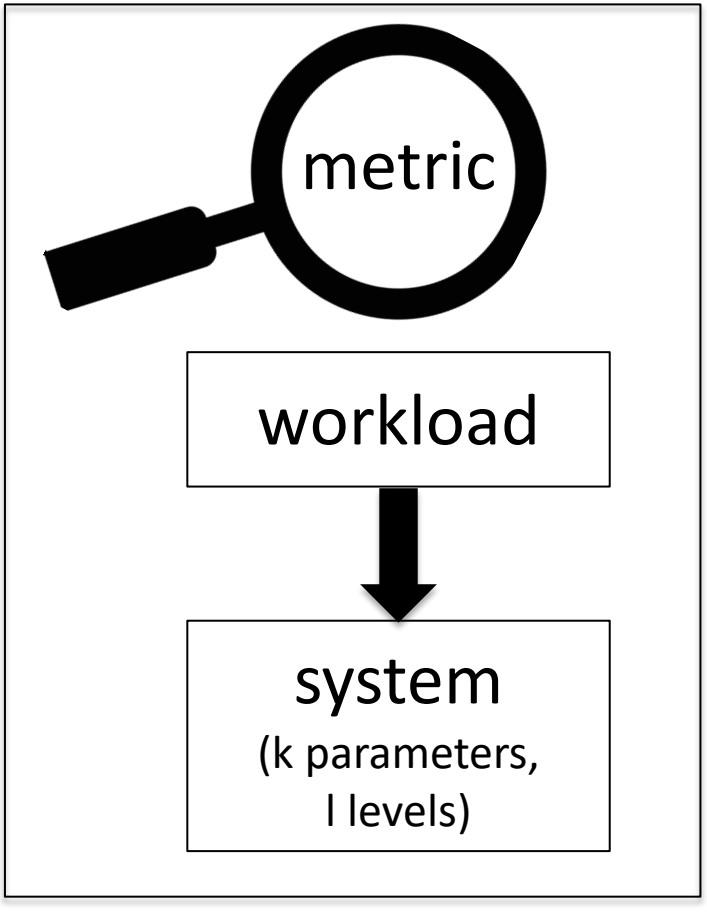
experiment

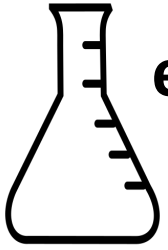
"a procedure undertaken to
make a discovery,
test a hypothesis,
or demonstrate a known fact."

google dictionary

start point / goal
before designing or
running any experiments

designing experiments





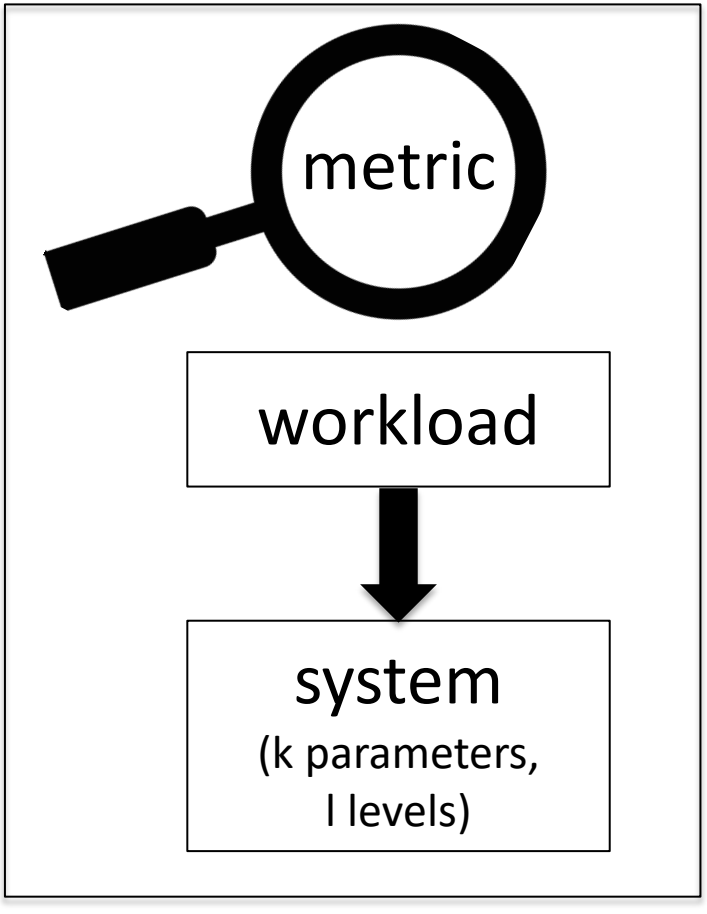
experiment

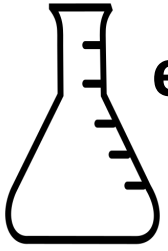
"a procedure undertaken to
make a discovery,
test a hypothesis,
or demonstrate a known fact."

google dictionary

would like to observe the throughput of system A?

designing experiments





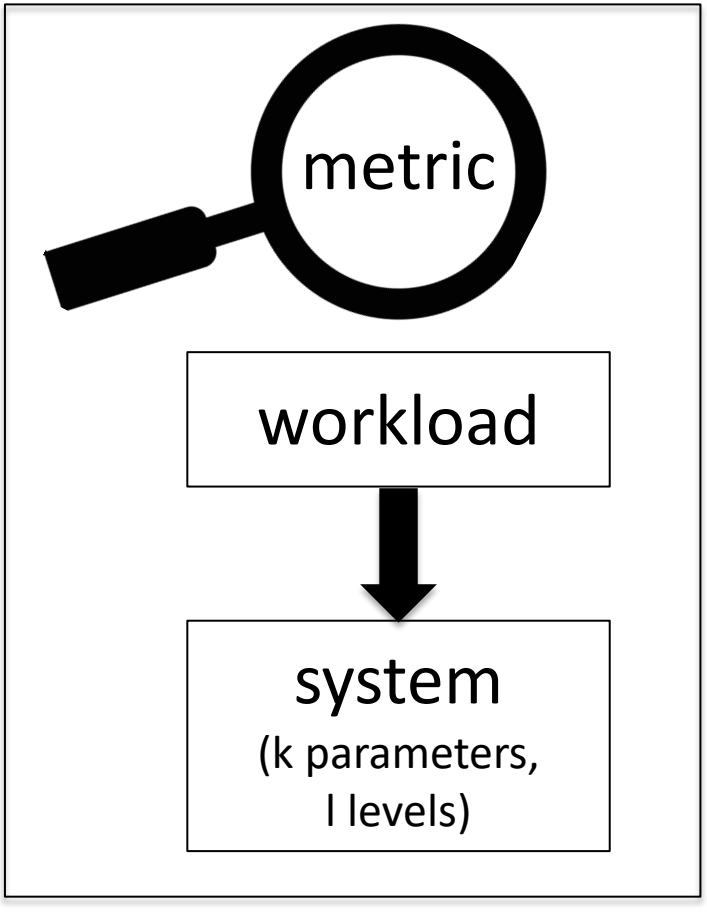
experiment

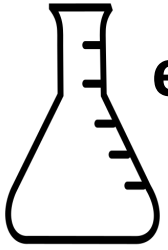
"a procedure undertaken to make a discovery, ***test a hypothesis****, or demonstrate a known fact."*

google dictionary

hypothesis to test:
system A has higher throughput than system B

designing experiments





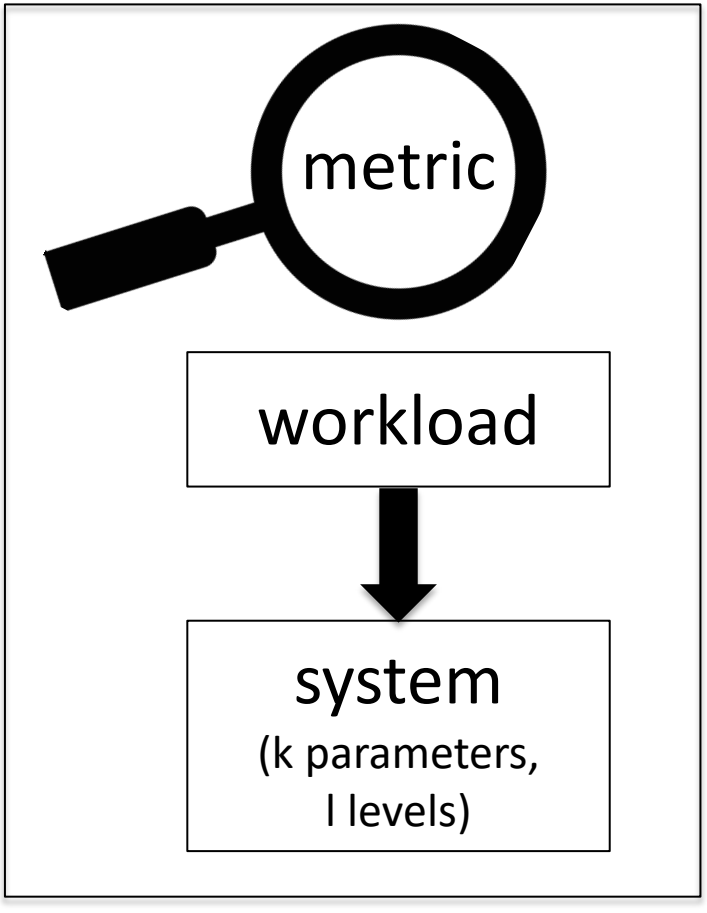
experiment

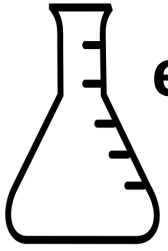
"a procedure undertaken to
make a discovery,
test a hypothesis,
*or **demonstrate a known fact**."*

google dictionary

it has been shown before that
system A has higher throughput
than system B,
would like to reproduce this result.

designing experiments





experiment

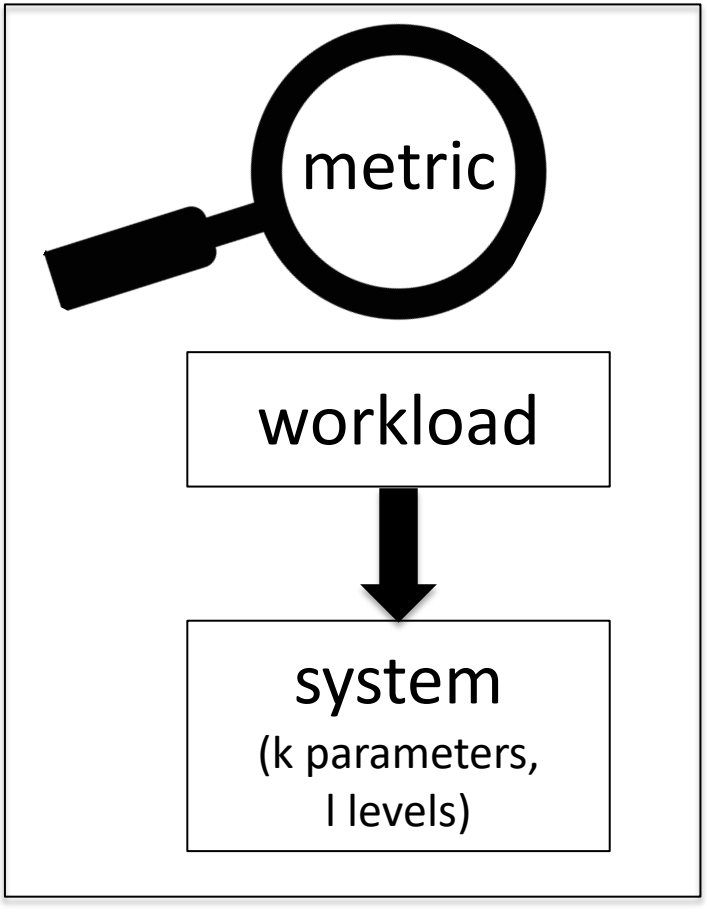
parameters: cores, memory
levels for cores: 1, 2, 4, 6, 8
levels for memory: 1GB, 10GB, 100GB

brute force ➔ $l_1 \times l_2 \times \ldots \times l_k$
5 x 3 = 15 experiments

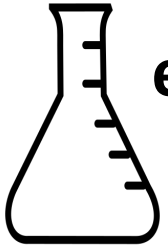
may get out of hand!

if you have infinite time & resources go ahead!

designing experiments



more common!



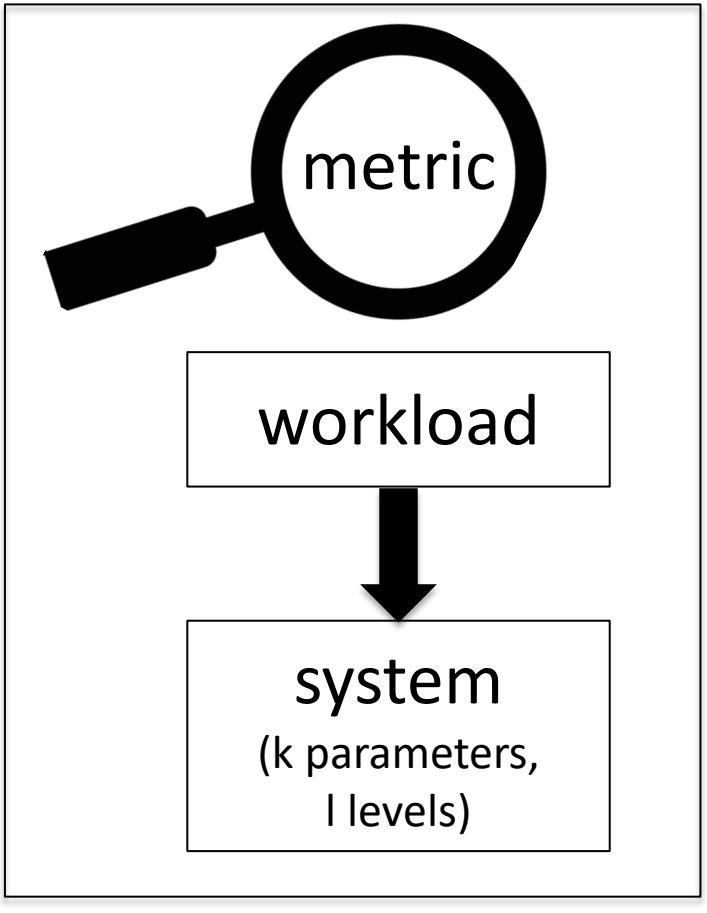
experiment

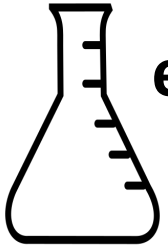
parameters: cores, memory
levels for cores: 1, 2, 4, 6, 8
levels for memory: 1GB, 10GB, 100GB

assume **a default level** for each parameter. **vary a single parameter's level** at a time, keep others at default.

default #cores = 1, memory = 100GB
5 + (3 - 1) = 7 experiments

designing experiments





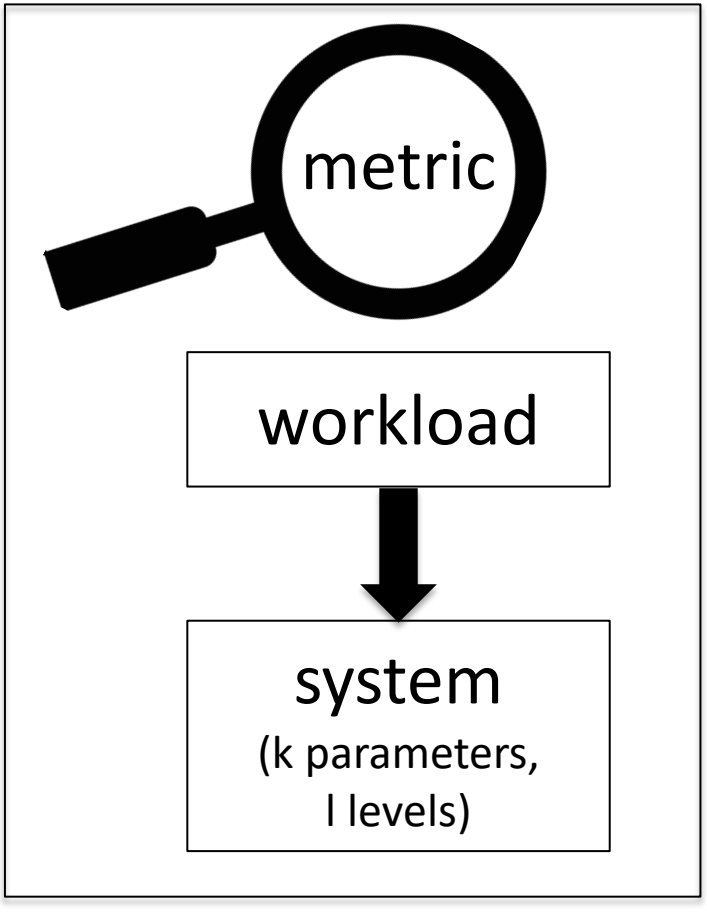
experiment

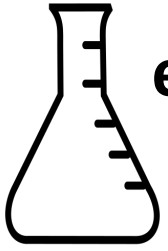
parameters: cores, memory
levels for cores: 1, 2, 4, 6, 8
levels for memory: 1GB, 10GB, 100GB

in practice, a combination of both is also common!

may want to test with several default values!

designing experiments





experiment

how many times to run a particular experiment?

statistics would help you here

in practice, do not run it just once & make sure standard deviation is low across different runs!
if not, you have an error to fix or a phenomenon to understand!

parameter with the most impact?

often more than one factor/parameter has effect

e.g., number of CPU cores, network connection speed, etc.

how to determine which one has biggest impact?

2^k factorial experiment

- for each factor (can be anything that affects our response variable), consider a ***low*** and ***high*** level.
- measure the system with all combinations (hence the 2^k)
- should be combined with repetitions

2^k factorial experiment

running your C program on different hardware

Throughput (requests per sec)	1GB Memory	16GB Memory
2MB Cache	32	68
8MB Cache	52	155

two factors: memory (x_A) and cache (x_B)

low level: $x_A = -1$

high level: $x_A = 1$

performance modeling

non-linear regression for performance (just a model!)

$$\text{throughput} = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

in our example:

$$32 = q_0 - q_A - q_B + q_{AB}$$

$$68 = q_0 + q_A - q_B - q_{AB}$$

$$52 = q_0 - q_A + q_B - q_{AB}$$

$$155 = q_0 + q_A + q_B + q_{AB}$$

after solving the system:

- $q_0 = 76.75$ (average of experiments)
- $q_A = 34.75$ (effect of memory)
- $q_B = 26.75$ (effect of cache)
- $q_{AB} = 16.75$ (effect of the interaction between the two)

- can use further statistics to understand the impact of a parameter on throughput
- will not get into more formulas in this course
- **this course will have a more computer systems focus**

in practice

you can use 2^k experiments as a starting point

not all 2^k experiments may be necessary,
    you can do fewer experiments

for parameters with higher impact,
    you may want to do >2 experiments for sensitivity analysis

do not forget to repeat your experiments to account for errors

summary

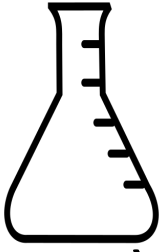
Performance analysis of computer systems is crucial for many scientific goals and requires understanding several layers of computer systems well.

There are several ways to perform such analysis using modeling, simulation, measurements.

Analysis also requires designing experiments.

Designing experiments require determining systems, workloads, metrics, experimental runs to reach the initial scientific goal.

performance analysis



experiment

6 decide on technique (model, simulate, measure)

8 actual experiment

1 **goal**

7 **workload**

2 **system**

4 (k parameters,

5 l levels)

9 **interpret results**

10 **present results**

keep the big picture in mind at all times!

next lecture

a case study

read “Data Partitioning on Chip Multiprocessors” &

analyze the experiments in this research paper

focusing on the following questions:

- what is the goal?
- what is/are the system(s)?
- what are the metrics?
- how many runs for the experiments?
- what are the good or bad points about the experiments?

also relevant for your first project!!