Benchmarking

Performance Engineering: Theory & Practice

Benchmarking

- Benchmarks
  - Performance tests that, when run **repeatedly** on the same platform, reliably produce very similar results
    - Execution time/Response time
    - Throughput
  - Synthetic benchmarks
    - Performance tests specifically developed to be **representative** of a broad class of common, computational problems
      - may include an element of randomness in order to discourage “benchmark engineering”
  - Micro-benchmarks
    - easy to run; but a narrow focus

Benchmarking

- Standardized benchmarks: usually synthetic benchmarks that are used to compare different platforms
  - For example:
    - as CPU architectures became more complex (μcode, cache, pipelining, OOO execution, RISC optimizing compilers, etc.), it became apparent that MIPS or clock speed alone (GHz) was inadequate as a basis for comparison
    - e.g., dhrystone
    - originally published ~ 1988
    - short (100 HLL C statements) synthetic benchmark program intended to be representative of system (i.e., integer) programming

Benchmarking

- LINPACK
  - Measures floating point performance
    - originally published ~ 1978
    - CPU-intensive: solves a randomly generated, dense n x n matrix, representing a set of linear equations
      - i.e., LINPACK 1000 : $n = 1000$
      - included in the Intel Math Kernel Library Benchmarks
      - High Performance Linpack (parallelism)
      - Supercomputer bragging rights regularly reported at Top500.org (link)
        - e.g., IBM Power System AC922 with 2 million cores

Benchmarking

- Problem:
  - Customer buys a system based (partially) on independent benchmark results
  - But, the installed system, running the customer's workload, however, does not measure up

- How *representative* is the benchmark workload of my actual workload?
  - Benchmarks measure something; the question is always, "Does it measure something useful and/or meaningful for me?"
  - How does the benchmark workload compare to my workload?
    - High Performance Conjugate Gradients (HPCG) compared to HP LINPACK
      - sparse matrix, Gauss-Seidel smoothing, etc.

Benchmarks proliferate!

- One size doesn’t fit all!

- SPEC (Standard Performance Evaluation Corporation)
  - generally, but not exclusively, CPU-intensive
  - portable, easy-to-run

- Transaction Processing Council
  - transaction-oriented Database workloads
  - throughput + cost/transaction

- Storage Performance Council
  - price/performance of IO subsystems

SPEC

- SPEC (Standard Performance Evaluation Corporation)
  - generally, but not exclusively, CPU-intensive
  - integer
  - floating point
  - vector graphics
  - HPC
  - energy consumption (Server Efficiency Rating Tool – SERT)
  - etc.

  - Only practical method to compare the performance of Intel vs. AMD vs. ARM vs. SPARC architectures

TPC

- Standardization effort inspired by Jim Gray's original Debit/Credit benchmark (~1985)
    - SPEC CPU-intensive benchmarks are not representative of many mission-critical commercial workloads
        - particularly Database back-ends
    - $/transaction measurements can be more important to many customers than peak transaction processing rates
    - so, TPC publishes *both*

    - Issues:
        - elaborate and relatively expensive to implement (compared to SPEC)
        - Vendors only publish results that are competitive !

TPC

- TPC-C: Order Entry
- TPC-DS: Decision Support (Big Data; read-only data warehouse)
- TPC-E: more complex OLTP
- TPC-VMS: TPC-C + TPC-DS + TPC-E + virtualization

- Issues:
  - Benchmarking vs. “Benchmarketing”
  - TPC $/transaction may depend more on IO subsystem performance than any other single factor!

  - motivation for the development of the SPC (Storage Performance Council) benchmarks

Storage benchmarking

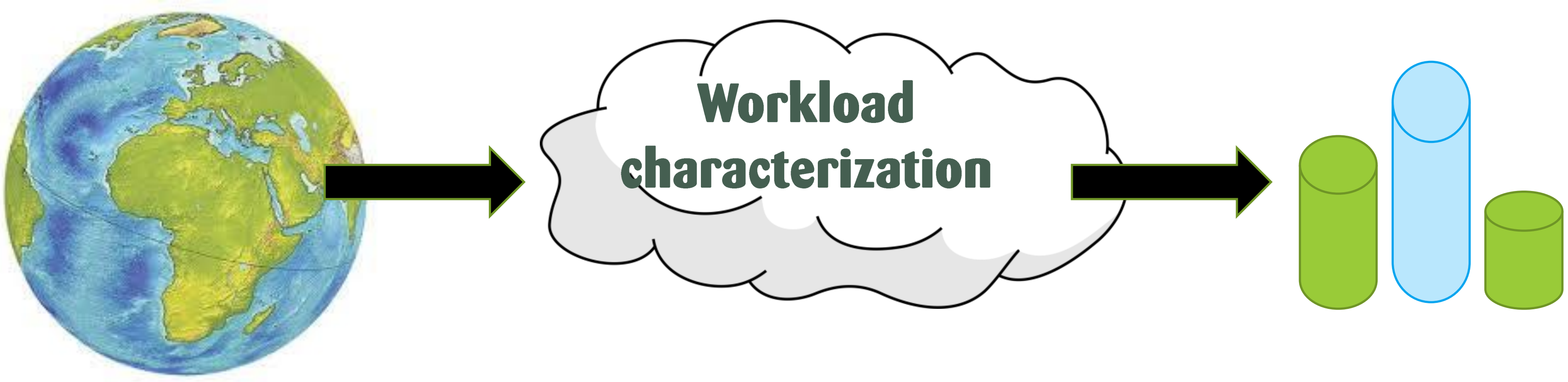
- Cost/performance of the storage subsystem being used is a major factor in TPC benchmarks
    - Often representing > 50% of the system cost
- Non-volatile RAM technology (SSDs) is game-changing!
    - No seek; no rotational delay
    - Bandwidth
    - performance of random Reads equivalent to sequential
    - Writes are usually significantly slower than Reads
- Example: see NetApp NVMe white paper

Benchmark Engineering

- When hardware/software developers build systems that are optimized to run specific benchmark workloads
- Some benchmark engineering is inevitable & quite innocent:
  - Developers evaluate new products in development based on execution of these standard benchmarking programs
- Where do you cross the line? From the TPC:
  - “*Specifically prohibited are benchmark systems, products, technologies or pricing…whose primary purpose is performance optimization of TPC benchmark results* **without any corresponding applicability to real-world** *applications and environments.*”

Why Benchmarks proliferate

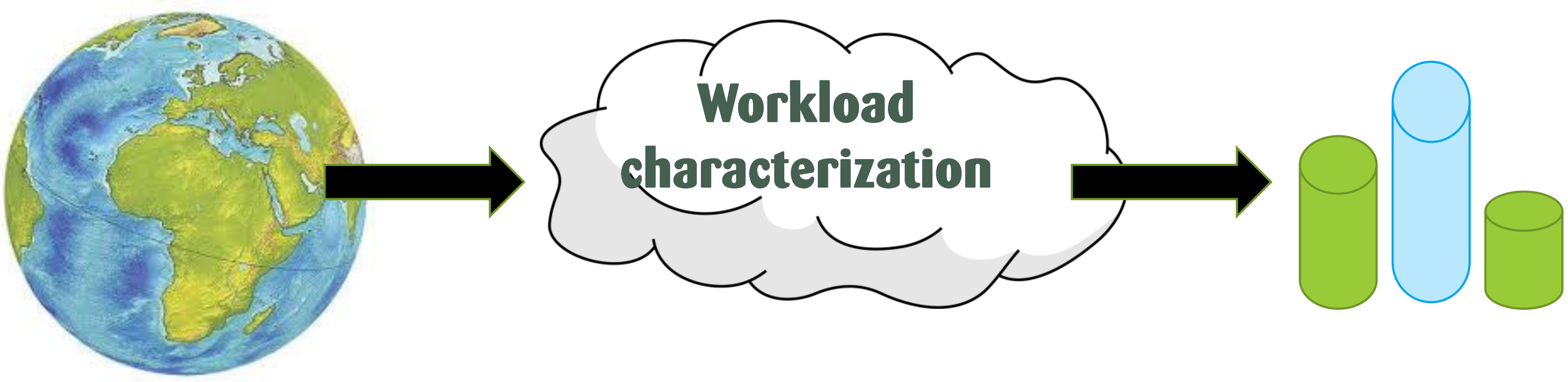
- Popular benchmarks attempt to encapsulate “important” real-world workloads
  - processor instruction mixes (integer vs. floating point)
  - representative scientific computing tasks (vector instructions)
  - transaction-processing (Create-Read-Update-Delete)
  - disk io (sequential, random, cache-friendliness)



Workload characterization

Workload characterization

- A statistical distillation process that is based on an (often implicit) underlying n-dimensional, **scalability model**
  - e.g.,
    - integer vs, scientific instruction mix
    - resource usage profiles for queueing models
    - CRUD operation mix



Workload characterization

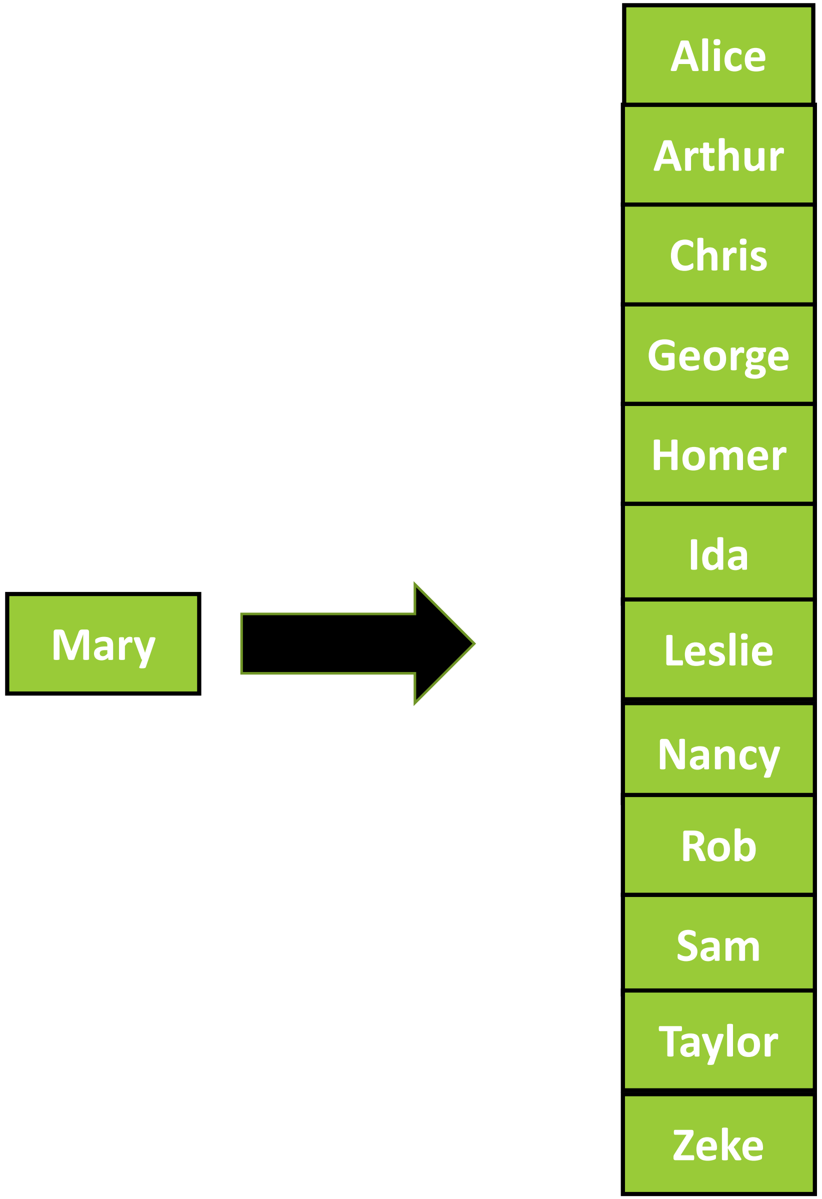
Scalability model

- A testable hypothesis that predicts the performance of a specific application workload on a designated computing platform
  - ▫ execution of repeatable benchmarks that encapsulate the workload are one way to test the viability of a scalability model

- For example, consider how various data structures perform standard CRUD operations:
  - ▫ arrays, lists, hash tables, binary trees, etc.

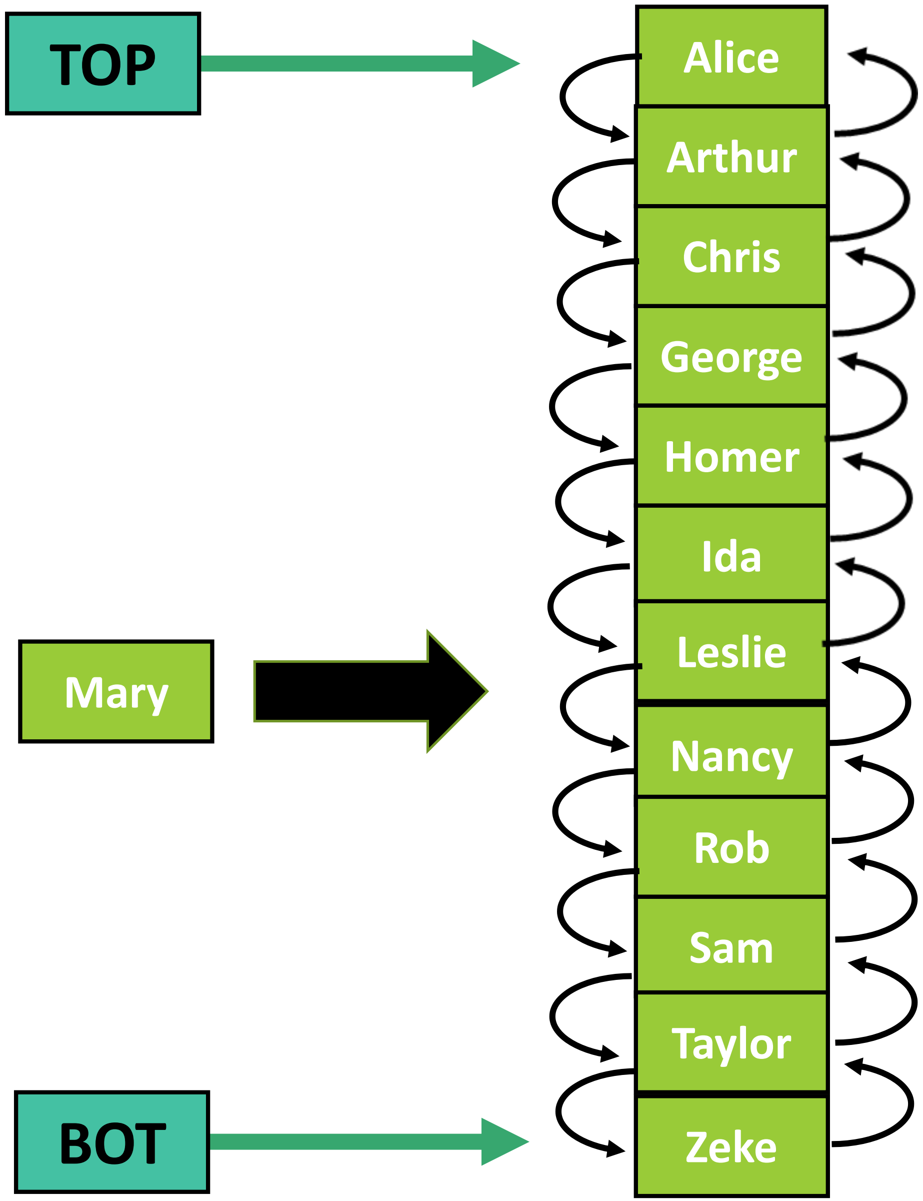
Scalability model

- Consider static arrays:
  - fast, efficient iteration
  - direct access using an integer indexer
  - sorted arrays support binary search
  - but **Inserts** and **Deletes** into an ordered array are problematic
  - especially as *n*, the size of the array, increases



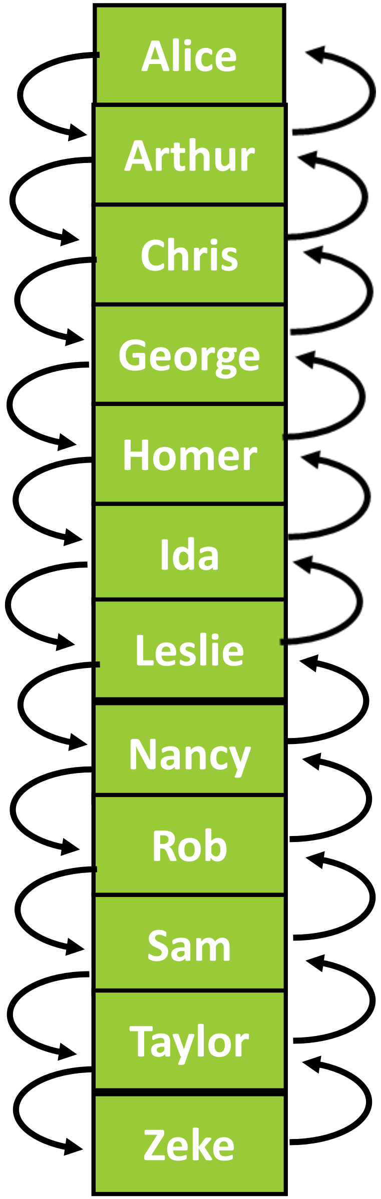
Scalability model

- Dynamic arrays and Lists
  - no direct indexing, but fast iteration by chasing Address pointers
- Sorted Lists
  - binary search
- Hash Tables
  - Key-Value-Pairs
  - collisions
- Binary trees
  - tree traversal
  - balanced binary trees



Scalability model for dynamic containers

N	Iterate	Insert	Search	Delete
10^3				
10^4				
10^5				
10^6	10%	15%	60%	15%
10^7				



Alice
Arthur
Chris
George
Homer
Ida
Leslie
Nancy
Rob
Sam
Taylor
Zeke

Homework

- Write a benchmark program to compare the scalability of standard Container (or Collection) classes in a familiar programming Framework
  - C++ (link)
  - Java (link)
  - C# (link)
- executing a synthetic CRUD workload

N	Iterate	Insert	Search	Delete
10^3				
10^4				
10^5				
10^6	10%	15%	60%	15%
10^7				

Homework

- Write a benchmark program to compare the scalability of standard Container (or Collection) classes in a familiar programming Framework
  - ▫ C++ (link)
  - ▫ Java (link)
  - ▫ C# (link)
  - ▫ etc.
- executing a synthetic CRUD workload

- Deliverables: due on 10/24
  1. Program listing
  2. Report results that demonstrate the benchmark is repeatable
  3. Analyze the results, reporting on the scalability of the various container classes, i.e.,
     - Meets expectations
     - Exceeds expectations
     - Fails to meet expectations

Questions

?

References

- Standard Performance Evaluation Corporation (SPEC)