Performance

Nick Sumner
wsumner@sfu.ca
Performance & Measurement

- Real development must manage resources
Performance & Measurement

- Real development must manage resources
  - Time
  - Memory
  - Open connections
  - VM instances
  - Energy consumption
  - ...

Performance & Measurement

- Real development must manage resources
  - Time
  - Memory
  - Open connections
  - VM instances
  - Energy consumption
  - ...

- Resource usage is one form of performance
  - *Performance* – a measure of nonfunctional behavior of a program
Performance & Measurement

- Real development must manage resources
  - Time
  - Memory
  - Open connections
  - VM instances
  - Energy consumption
  - ...

- Resource usage is one form of performance
  - **Performance** – a measure of nonfunctional behavior of a program

- **We often need to assess performance or a change in performance**
  - Data Structure A  vs  Data Structure B
Performance & Measurement

- Real development must manage resources
  - Time
  - Memory
  - Open connections
  - VM instances
  - Energy consumption
  - ...

- Resource usage is one form of performance
  - *Performance* – a measure of nonfunctional behavior of a program

- We often need to assess performance or a change in performance

  Data Structure A  **vs**  Data Structure B

How would you approach this in a data structures course?
Performance & Measurement

- Performance assessment is deceptively hard

[Demo/Exercise]
Performance & Measurement

- Performance assessment is deceptively hard
  - Modern systems involve complex actors
Performance & Measurement

- **Performance assessment is deceptively hard**
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
Performance & Measurement

- **Performance assessment is deceptively hard**
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
  - Even with the best intentions we can deceive ourselves
Performance & Measurement

- Performance assessment is deceptively hard
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
  - Even with the best intentions we can deceive ourselves

- Good performance evaluation should be rigorous & scientific
Performance & Measurement

- Performance assessment is deceptively hard
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
  - Even with the best intentions we can deceive ourselves

- Good performance evaluation should be rigorous & scientific
  - The same process applies in development as in good research
Performance & Measurement

- Performance assessment is deceptively hard
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
  - Even with the best intentions we can deceive ourselves

- Good performance evaluation should be rigorous & scientific
  - The same process applies in development as in *good* research
    1) Clear claims
    2) Clear evidence
    3) Correct reasoning from evidence to claims
Performance & Measurement

- Performance assessment is deceptively hard
  - Modern systems involve complex actors
  - Theoretical models may be too approximate
  - Even with the best intentions we can deceive ourselves

- Good performance evaluation should be rigorous & scientific
  - The same process applies in development as in good research
    1) Clear claims
    2) Clear evidence
    3) Correct reasoning from evidence to claims
  - And yet this is challenging to get right!
Performance and Measurement

Several facets:

- **Speed / Running time**
  - The total time required (latency?)

- **Throughput**
  - Pages/Transactions per second, bytes per second

- **Responsiveness**
  - UI response time, server response time at peak load

- **Memory Consumption**
  - Peak memory consumption

- ...
Measurement

- So how can we measure these?
Measurement

- So how can we measure these?
  - Idea: run the test suite and measure the resource in question
So how can we measure these?
- Idea: run the test suite and measure the resource in question
- How well does this capture system level performance?
- “ ” low level performance?
Measurement

- So how can we measure these?
  - Idea: run the test suite and measure the resource in question
  - How well does this capture system level performance?
  - “ ” low level performance?

- A functionality based test suite will not capture performance concerns!
  - Design tests that specifically target performance objectives
• **So how can we measure these?**
  - Idea: run the test suite and measure the resource in question
  - How well does this capture system level performance?
  - “ ” low level performance?

• **A functionality based test suite will not capture performance concerns!**
  - Design tests that specifically target performance objectives

How? What should the tests capture?
Benchmarking

• We must reason rigorously about performance during assessment, investigation, & improvement
Benchmarks

- We must reason rigorously about performance during assessment, investigation, & improvement
- *Assessing performance is done through benchmarking*
Benchmarking

- We must reason rigorously about performance during assessment, investigation, & improvement

- Assessing performance is done through benchmarking
  - *Microbenchmarks*
    - Focus on cost of an operation in isolation
    - Can help identify core performance details & explain causes
Benchmarking

- We must reason rigorously about performance during assessment, investigation, & improvement

- **Assessing performance is done through benchmarking**
  - *Microbenchmarks*
    - Focus on cost of an operation in isolation
    - Can help identify core performance details & explain causes
  - *Macrobenchmarks*
    - Real world system performance
Benchmarks

- We must reason rigorously about performance during assessment, investigation, & improvement
- **Assessing performance is done through benchmarking**
  - *Microbenchmarks*
    - Focus on cost of an operation in isolation
    - Can help identify core performance details & explain causes
  - *Macrobenchmarks*
    - Real world system performance
- Workloads (inputs) must be chosen carefully either way.
  - representative, pathological, scenario driven, ...
Suppose we want to run a microbenchmark

```java
startTime = getCurrentTimeInSeconds();
doWorkloadOfInterest();
endTime = getCurrentTimeInSeconds();
reportResult(endTime - startTime);
```
Suppose we want to run a microbenchmark:

```java
startTime = getCurrentTimeInSeconds();
doWorkloadOfInterest();
endTime = getCurrentTimeInSeconds();
reportResult(endTime - startTime);
```

What possible issues do you observe?
Benchmarking

- Suppose we want to run a microbenchmark

```java
startTime = getCurrentTimeInSeconds();
doWorkloadOfInterest();
endTime = getCurrentTimeInSeconds();
reportResult(endTime - startTime);
```

- Granularity of measurement
- Warm up effects
- Nondeterminism
- Size of workload
- System interference
- Frequency scaling?
- Interference of other workloads?
- Alignment?
Benchmarking

- **Granularity & Units**
  - Why is granularity a problem?
  - What are alternatives to `getCurrentTimeInSeconds()`?
BENCHMARKING

- Granularity & Units
  - Why is granularity a problem?
  - What are alternatives to `getCurrentTimeInSeconds()`?
  - What if I want to predict performance on a different machine?
Benchmarking

- **Granularity & Units**
  - Why is granularity a problem?
  - What are alternatives to `getCurrentTimeInSeconds()`?
  - What if I want to predict performance on a different machine?
    - Using *cycles* instead of wall clock time can be useful, but has its own limitations
Benchmarking

- Warm up time
  - Why is warm up time necessary in general?
Benchmarking

- Warm up time
  - Why is warm up time necessary *in general*?
  - Why is it especially problematic for systems like Java?
Benchmarking

- **Warm up time**
  - Why is warm up time necessary *in general*?
  - Why is it especially problematic for systems like Java?
  - How can we modify our example to facilitate this?
Benchmarking

- Warm up time
  - Why is warm up time necessary \textit{in general}?
  - Why is it especially problematic for systems like Java?
  - How can we modify our example to facilitate this?

```java
for (...) doWorkloadOfInterest();
startTime = getCurrentTimeInSeconds();
doWorkloadOfInterest();
endTime = getCurrentTimeInSeconds();
reportResult(endTime – startTime);
```
Benchmarking

- **Nondeterministic behavior**
  - **Will** `getCurrentTimeInSeconds()` **always return the same number?**

  *Why/why not?*
Benchmarking

- Nondeterministic behavior
  - Will `getCurrentTimeInSeconds()` always return the same number?
  - So what reflects a meaningful result?
    - Hint: The Law of Large Numbers!
Benchmarking

• **Nondeterministic behavior**
  - Will `getCurrentTimeInSeconds()` always return the same number?
  - So what reflects a *meaningful* result?
    • Hint: The Law of Large Numbers!

• **By running the same test many times, the arithmetic mean will converge on the expected value**
Benchmarking

- Nondeterministic behavior
  - Will `getCurrentTimeInSeconds()` always return the same number?
  - So what reflects a meaningful result?
    - Hint: The Law of Large Numbers!

- By running the same test many times, the arithmetic mean will converge on the expected value

Is this always what you want?
Benchmarking

- A revised (informal) approach:

```java
for (...) doWorkloadOfInterest();
startTime = getCurrentTimeInNanos();
for (...) doWorkloadOfInterest();
endTime = getCurrentTimeInNanos();
reportResult(endTime - startTime);
```
Benchmarking

- A revised (informal) approach:

```java
for (...) doWorkloadOfInterest();
startTime = getCurrentTimeInNanos();
for (...) doWorkloadOfInterest();
endTime = getCurrentTimeInNanos();
reportResult(endTime - startTime);
```

- This still does not solve everything
  - Frequency scaling?
  - Interference of other workloads?
  - Alignment?
Now we have a benchmark, how do we interpret/report it?
- We must *compare*
Benchmarking

- Now we have a benchmark, how do we interpret/report it?
  - We must *compare*
    - Benchmark vs expectation/mental model
    - Different solutions
    - Over time
Benchmarking

- Now we have a benchmark, how do we interpret/report it?
  - We must **compare**
    - Benchmark vs expectation/mental model
    - Different solutions
    - Over time
  - Results are often normalized against the baseline
Now we have a benchmark, how do we interpret/report it?

- We must compare
- We must remember results are statistical
Benchmarking

- Now we have a benchmark, how do we interpret/report it?
  - We must *compare*
  - We must remember results are *statistical*
    - Show the distribution (e.g. violin plots)
Benchmarking

- Now we have a benchmark, how do we interpret/report it?
  - We must *compare*
  - We must remember results are *statistical*
    - Show the distribution (e.g. violin plots)
    - Summarize the distribution (e.g. mean and confidence intervals, box & whisker)
Benchmarking

- A benchmark suite comprises multiple benchmarks
Benchmarking

- A benchmark suite comprises multiple benchmarks
- Now we have multiple results, how should we consider them?

![Bar chart showing benchmark results for T1 to T6, old vs new versions.]
Benchmarking

- A benchmark suite comprises multiple benchmarks
- Now we have multiple results, how should we consider them?
  - 2 major scenarios
    - Hypothesis testing
      - Is solution A different than B?
Benchmarking

- A benchmark suite comprises multiple benchmarks

- Now we have multiple results, how should we consider them?
  - 2 major scenarios
    - *Hypothesis testing*
      - Is solution A different than B?
      - You can use ANOVA

---

**Graph:**

- **Old**
- **New**

- **T1**
- **T2**
- **T3**
- **T4**
- **T5**
- **T6**
Benchmarking

- A benchmark suite comprises multiple benchmarks
- Now we have multiple results, how should we consider them?
  - 2 major scenarios
    - *Hypothesis testing*
    - *Summary statistics*
A benchmark suite comprises multiple benchmarks

Now we have multiple results, how should we consider them?
- 2 major scenarios
  - *Hypothesis testing*
  - *Summary statistics*
    - Condensing a suite to a single number
    - Intrinsically lossy, but can still be useful
Benchmarking

- A benchmark suite comprises multiple benchmarks

- Now we have multiple results, how should we consider them?
  - 2 major scenarios
    - *Hypothesis testing*
    - *Summary statistics*
      - Condensing a suite to a single number
      - Intrinsically lossy, but can still be useful

![Bar chart comparing Old vs New results across T1 to T6]
Summary Statistics

Averages of $r_1, r_2, \ldots, r_N$

- Many ways to measure *expectation* or *tendency*
Summary Statistics

Averages of \( r_1, r_2, \ldots, r_N \)

- Many ways to measure *expectation* or *tendency*
- Arithmetic Mean

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]
Summary Statistics

Averages of $r_1, r_2, ..., r_N$

- Many ways to measure *expectation* or *tendency*
- Arithmetic Mean
  \[ \frac{1}{N} \sum_{i=1}^{N} r_i \]
- Harmonic Mean
  \[ \frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}} \]
Summary Statistics

Averages of \( r_1, r_2, \ldots, r_N \)

- Many ways to measure *expectation* or *tendency*
- Arithmetic Mean
  \[
  \frac{1}{N} \sum_{i=1}^{N} r_i
  \]
- Harmonic Mean
  \[
  \frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}}
  \]
- Geometric Mean
  \[
  \sqrt[N]{ \prod_{i=1}^{N} r_i }
  \]
Summary Statistics

Averages of \( r_1, r_2, \ldots, r_N \)

- Many ways to measure expectation or tendency
- Arithmetic Mean
  \[
  \frac{1}{N} \sum_{i=1}^{N} r_i
  \]
- Harmonic Mean
  \[
  \frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}}
  \]
- Geometric Mean
  \[
  \sqrt[N]{\prod_{i=1}^{N} r_i}
  \]

Each type means something different and has valid uses
Summary Statistics

• **Arithmetic Mean**
  – Good for reporting averages of numbers that mean the same thing

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]
Summary Statistics

- **Arithmetic Mean**
  - Good for reporting averages of numbers that mean the same thing
  - Used for computing *sample means*

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]
Summary Statistics

- **Arithmetic Mean**
  - Good for reporting averages of numbers that mean the same thing
  - Used for computing sample means
  - e.g. Timing the same workload many times

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]
Summary Statistics

- **Arithmetic Mean**
  - Good for reporting averages of numbers that mean the same thing
  - Used for computing sample means
  - e.g. Timing the same workload many times

\[
E(\text{time}) = \frac{1}{N} \sum_{i=1}^{N} r_i
\]

Handling Nondeterminism

```python
for (x in 0 to 4)
    times[x] = doWorkloadOfInterest();

E(\text{time}) = \text{arithmean}(\text{times})
```
Summary Statistics

- Arithmetic Mean
  - Good for reporting averages of numbers that mean the same thing
  - Used for computing sample means
  - e.g. Timing the same workload many times

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]

- Harmonic Mean
  - Good for reporting rates

\[
\frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}}
\]
Summary Statistics

- **Arithmetic Mean**
  - Good for reporting averages of numbers that mean the same thing
  - Used for computing sample means
  - e.g. Timing the same workload many times

- **Harmonic Mean**
  - Good for reporting *rates*
  - e.g. Required throughput for a set of tasks

$$\frac{1}{N} \sum_{i=1}^{N} r_i$$

$$\frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}}$$
Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
- throughput(t1) = 10 pages/sec
- throughput(t2) = 20 pages/sec
- throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?

- Good for reporting rates
- e.g. Required throughput for a set of tasks

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]

\[
\frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}}
\]
Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
- throughput(t1) = 10 pages/sec
- throughput(t2) = 20 pages/sec
- throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?
Arithmetic = 16.7 p/s

- Good for reporting rates
- e.g. Required throughput for a set of tasks
Summary Statistics

Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
  throughput(t1) = 10 pages/sec
  throughput(t2) = 20 pages/sec
  throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?
Arithmetic = 16.7 p/s     Harmonic = 15 p/s

- Good for reporting rates
- e.g. Required throughput for a set of tasks

\[ \frac{1}{N} \sum_{i=1}^{N} r_i \]
Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
  throughput(t1) = 10 pages/sec
  throughput(t2) = 20 pages/sec
  throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?
Arithmetic = 16.7 p/s     Harmonic = 15 p/s
120/16.7 = 7.2          120/15 = 8

- Good for reporting rates
- e.g. Required throughput for a set of tasks
Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
- throughput(t1) = 10 pages/sec
- throughput(t2) = 20 pages/sec
- throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?
- Arithmetic = 16.7 p/s
- Harmonic = 15 p/s

Identifies the constant rate required for the same time

\[
\frac{1}{N} \sum_{i=1}^{N} r_i
\]
Summary Statistics

Given tasks t1, t2, & t3 serving 40 pages each:
  - throughput(t1) = 10 pages/sec
  - throughput(t2) = 20 pages/sec
  - throughput(t3) = 20 pages/sec

What is the average throughput? What should it mean?
- Arithmetic = \( \frac{120}{16.7} = 7.2 \) p/s
- Harmonic = \( \frac{120}{15} = 8 \) p/s

Identifies the constant rate required for the same time

CAVEAT: If the size of each workload changes, a weighted harmonic mean is required!
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]

Any idea why it may be useful here? (A bit of a thought experiment)
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]

What happens to the arithmetic mean?

Old

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>halved</td>
</tr>
</tbody>
</table>

New 2

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
</table>
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[\text{N}]{\prod_{i=1}^{\text{N}} r_i}
\]

The (non) change to T1 dominates any behavior for T2!
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

Geometric:

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

Geometric:

\[ \sqrt[N]{\prod_{i=1}^{N} r_i} \]

- Old: \[ \sqrt{r_1 \times r_2} \]
- Old: \[ \sqrt{r_1 \times \left(\frac{1}{2} r_2\right)} \]
- New 1: \[ \sqrt{r_1 \times \left(\frac{1}{2} r_2\right)} \]
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt{\frac{1}{N} \prod_{i=1}^{N} r_i}
\]

Geometric:

- Old: \(\sqrt{r_1 \times r_2}\)
- New 1: \(\sqrt{r_1 \times \left(\frac{1}{2} r_2\right)}\)
- New 2: \(\sqrt{\left(\frac{1}{2} r_1\right) \times r_2}\)
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks

\[
\sqrt[N]{\prod_{i=1}^{N} r_i}
\]

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>Old</td>
</tr>
</tbody>
</table>

Geometric:

\[
\sqrt{r_1 \times r_2}
\]

Old

\[
\sqrt{r_1 \times \left(\frac{1}{2} r_2\right)}
\]

New 1

\[
\sqrt{\frac{1}{2} \times r_1 \times r_2} = \sqrt{\left(\frac{1}{2} r_1\right) \times r_2}
\]

New 2
Summary Statistics

- **Geometric Mean**
  - Good for reporting results that mean different things
  - e.g. Timing results across *many different* benchmarks
  - A 10% difference in any benchmark affects the final value the same way

\[ \sqrt[N]{\prod_{i=1}^{N} r_i} \]
Summary Statistics

• **Geometric Mean**
  – Good for reporting results that mean different things
  – e.g. Timing results across *many different* benchmarks
  – A 10% difference in any benchmark affects the final value the same way

Note: It doesn't have an *intuitive* meaning! It does provide a balanced *score* of performance.

Summary Statistics

- Remember the distributions
  - Measurement is inherently nondeterministic
  - Every measurement is a sample from a probability distribution
Summary Statistics

- Remember the distributions
  - Measurement is inherently nondeterministic
  - Every measurement is a sample from a probability distribution

- Do these mean the same thing?
Summary Statistics

- Remember the distributions
  - Measurement is inherently nondeterministic
  - Every measurement is a sample from a probability distribution

- Do these mean the same thing?
  - You cannot ignore the spread of data
  - You at least need to account for the *sample standard deviation*
Summary Statistics

- Remember the distributions
  - Measurement is inherently nondeterministic
  - Every measurement is a sample from a probability distribution

- Do these mean the same thing?
  - You cannot ignore the spread of data
  - You at least need to account for the sample standard deviation

- *Recall* that the standard deviation provides a notion of the spread
  - Can be used to establish confidence in the mean
  - If it is large (1) you may have methodological error (2) you may need more data
Summary Statistics

- Remember the distributions
  - Measurement is inherently nondeterministic
  - Every measurement is a sample from a probability distribution

- Do these mean the same thing?
  - You cannot ignore the spread of data
  - You at least need to account for the sample standard deviation

- Recall that the standard deviation provides a notion of the spread
  - Can be used to establish confidence in the mean
  - If it is large (1) you may have methodological error (2) you may need more data

- More rigorously, consider
  - Confidence intervals, T-tests, & ANOVA
Benchmarking

- In practice applying good benchmarking & statistics is made easier via frameworks
  - Google benchmark (C & C++)
  - Google Caliper (Java)
  - Nonius
  - Celero
  - Easybench
  - Pyperf
  - ...

Perf & event profiling

- Sometimes low-level architectural effects determine the performance
  - Cache misses
  - Misspeculations
  - TLB misses
Perf & event profiling

- Sometimes low-level architectural effects determine the performance
  - Cache misses
  - Misspeculations
  - TLB misses

  How well does sample based profiling work for these?
Perf & event profiling

- Sometimes low-level architectural effects determine the performance
  - Cache misses
  - Misspeculations
  - TLB misses

  How well does sample based profiling work for these?

- Instead, we can leverage low level system counters via tools like perf
Perf & event profiling

- Sometimes low-level architectural effects determine the performance
  - Cache misses
  - Misspeculations
  - TLB misses

  How well does sample based profiling work for these?

- Instead, we can leverage low level system counters via tools like perf

  ```
  perf stat -e <events> -g <command>
  perf record -e <events> -g <command>
  perf report
  perf list
  ```
Perf & event profiling

- Sometimes low-level architectural effects determine the performance
  - Cache misses
  - Misspeculations
  - TLB misses

How well does sample based profiling work for these?

- Instead, we can leverage low level system counters via tools like perf

```
perf stat -e <events> -g <command>
perf record -e <events> -g <command>
perf report
perf list
```

Events like

```
task-clock, context-switches, cpu-migrations, page-faults, cycles, instructions, branches, branch-misses, cache-misses, cycle_activity.stalls_total
```
Improving real world algorithmic performance comes from recognizing the *interplay* between *theory* and *hardware*.
Optimizing Algorithms

- Improving real world algorithmic performance comes from recognizing the **interplay** between **theory** and **hardware**

- **Hybrid algorithms**
  - Constants matter. Use thresholds to select algorithms.
Optimizing Algorithms

- Improving real world algorithmic performance comes from recognizing the *interplay* between *theory* and *hardware*.

- **Hybrid algorithms**
  - Constants matter. Use thresholds to select algorithms.
  - Use general $N \log N$ sorting for $N$ above 300 [Alexandrescu 2019]
Optimizing Algorithms

- Improving real world algorithmic performance comes from recognizing the **interplay** between **theory** and **hardware**

- Hybrid algorithms
  - Constants matter. Use thresholds to select algorithms.
  - Use general N logN sorting for N above 300 [Alexandrescu 2019]

- **Caching & Precomputing**
Optimizing Algorithms

- Improving real world algorithmic performance comes from recognizing the *interplay* between *theory* and *hardware*.

- **Hybrid algorithms**
  - Constants matter. Use thresholds to select algorithms.
  - Use general $N \log N$ sorting for $N$ above 300 [Alexandrescu 2019]

- **Caching & Precomputing**
  - If you will reuse results, save them and avoid recomputing.
Optimizing Algorithms

- Improving real world algorithmic performance comes from recognizing the *interplay* between *theory* and *hardware*.

- Hybrid algorithms
  - Constants matter. Use thresholds to select algorithms.
  - Use general $N \log N$ sorting for $N$ above 300 [Alexandrescu 2019]

- **Caching & Precomputing**
  - If you will reuse results, save them and avoid recomputing
  - If all possible results are compact, just compute a table up front
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts
- Classic asymptotic complexity less useful in practice
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts
- Classic asymptotic complexity less useful in practice
  - It uses an *abstract machine model* that is too approximate!
Optimizing Algorithms

• Better performance modeling & algorithms
  – The core approaches we use have not adapted to changing contexts

• Classic asymptotic complexity less useful in practice
  – It uses an abstract machine model that is too approximate!
  – Constants and artifacts of scale can actually dominate the real world performance
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance

A *uniform cost model* throws necessary information away
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts
- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts for scale can actually dominate the real-world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shape of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts
- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts for scalability actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
  - Cache oblivious algorithms & data structures
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
  - Cache oblivious algorithms & data structures
    - Similar to I/O, but agnostic to block size
Optimizing Algorithms

- Better performance modeling & algorithms
  - The core approaches we use have not adapted to changing contexts

- Classic asymptotic complexity less useful in practice
  - It uses an abstract machine model that is too approximate!
  - Constants and artifacts of scale can actually dominate the real world performance
  - We want modeling & algorithms that account for artifacts like: memory, I/O, consistency & speculation, shapes of workloads

- Alternative approaches
  - I/O complexity, I/O efficiency and cache awareness
  - Cache oblivious algorithms & data structures
  - Parameterized complexity
Reasoning rigorously about performance is challenging
Summary

- Reasoning rigorously about performance is challenging
- **Good tooling can allow you to investigate performance well**