Real Logic

ACCELERATING SOFTWARE

# Rectangles All The Way Down

Martin Thompson - @mjpt777

*“The most amazing achievement of the computer software industry is its continuing cancellation of the steady and staggering gains made by the computer hardware industry.”*

*- Henry Peteroski*

# Fundamental Laws

# CPU Performance – Memory Lane

- *"Transistor density doubles every year"*

  *- Gordon Moore*

1965

# CPU Performance – Memory Lane

- *“Transistor density doubles every 2 years”*
  *- Gordon Moore*

- *“Transistor density doubles every year”*
  *- Gordon Moore*

**1965** **1975**

# CPU Performance – Memory Lane

- *"CPUs double in speed every 18 months"*
  *- David House*
- *"Transistor density doubles every 2 years"*
  *- Gordon Moore*
- *"Transistor density doubles every year"*
  *- Gordon Moore*

1965

1975

# CPU Performance – Memory Lane

*"The free lunch is over:"*
*- Herb Sutter*

*"CPUs double in speed every 18 months"*
*- David House*

*"Transistor density doubles every 2 years"*
*- Gordon Moore*

*"Transistor density doubles every year"*
*- Gordon Moore*

1965 1975 2005

# CPU Performance – Memory Lane

*Retirement of Tick Tock*

*- Intel*

*“The free lunch is over:”*

*- Herb Sutter*

*“CPUs double in speed every 18 months”*

*- David House*

*“Transistor density doubles every 2 years”*

*- Gordon Moore*

*“Transistor density doubles every year”*

*- Gordon Moore*

1965 1975 2005 2016

# CPU Performance – Memory Lane

*Spectre & Meltdown*
*- Google*

*Retirement of Tick Tock*
*- Intel*

*“The free lunch is over:”*
*- Herb Sutter*

*“CPUs double in speed every 18 months”*
*- David House*

*“Transistor density doubles every 2 years”*
*- Gordon Moore*

*“Transistor density doubles every year”*
*- Gordon Moore*

1965 1975 2005 2016 2018

# Concurrency & Parallelism


Road Safety Week
GRADUATE

# Universal Scalability Law (USL)

`C(N) = N / (1 + α(N – 1) + ((β* N) * (N – 1)))`

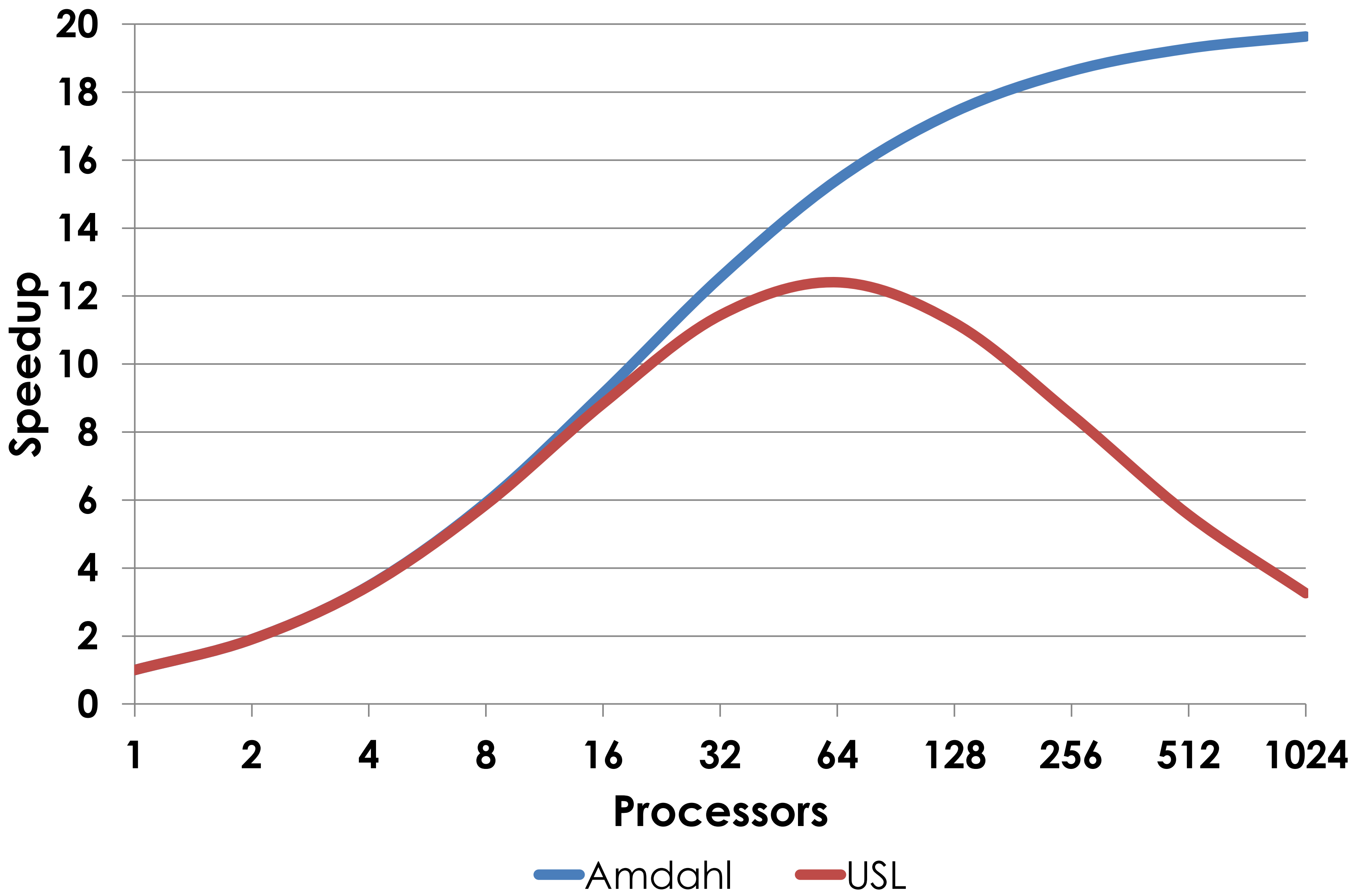
**C** = capacity or throughput

**N** = number of processors

**α** = ***contention*** penalty

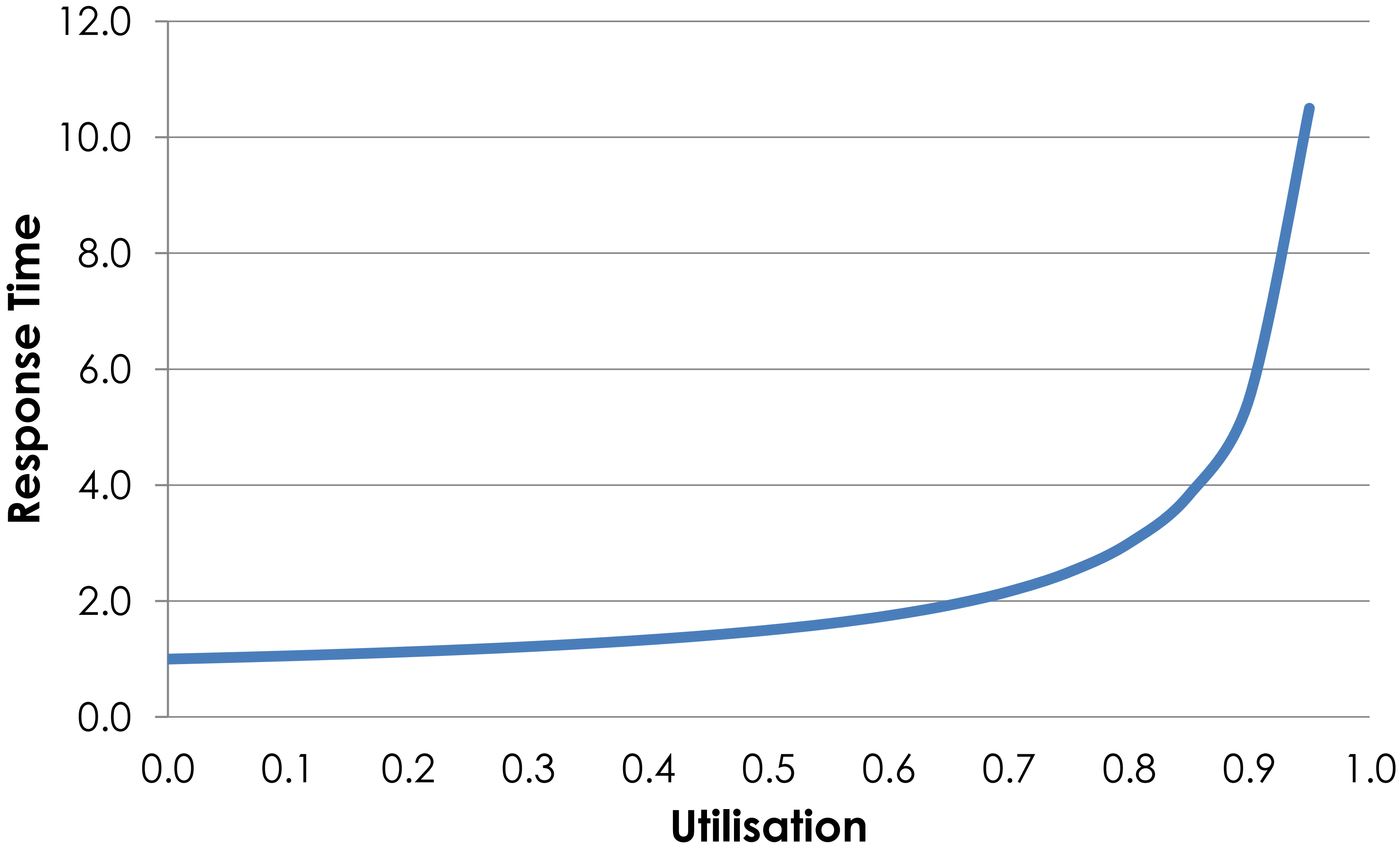
**β** = ***coherence*** penalty

# Universal Scalability Law (USL)



# If concurrency is so difficult then what else can we do?

# Queueing Theory



## Queueing Theory

$$r = s(2 - \rho) / 2(1 - \rho)$$

**r** = mean response time

**s** = service time

$\rho$ = utilisation

***Note:*** $\rho = \lambda * s$

# Little's Law

$$L = \lambda W$$

WIP = Throughput * Cycle Time

# Little’s Law

$$L = \lambda W$$

**WIP = Throughput * Cycle Time**

**<u>Bandwidth Delay Product</u>:**

**Bytes in flight = Bandwidth * Latency**

# Little's Law

$$L = \lambda W$$

**WIP = Throughput * Cycle Time**

**<u>Bandwidth Delay Product</u>:**

**Bytes in flight = Bandwidth * Latency**

**80 bytes / 100ns = 800 MB/s :10 LFBs**

# Memory

# Are all memory operations equal?

# Sequential Access

# -

# Average time in ns/op to sum all longs in a 1GB array?

# Access Pattern Benchmark

```
Benchmark                    Score     Error   Units
=====================================================
sequential                   0.832  ± 0.006   ns/op
```

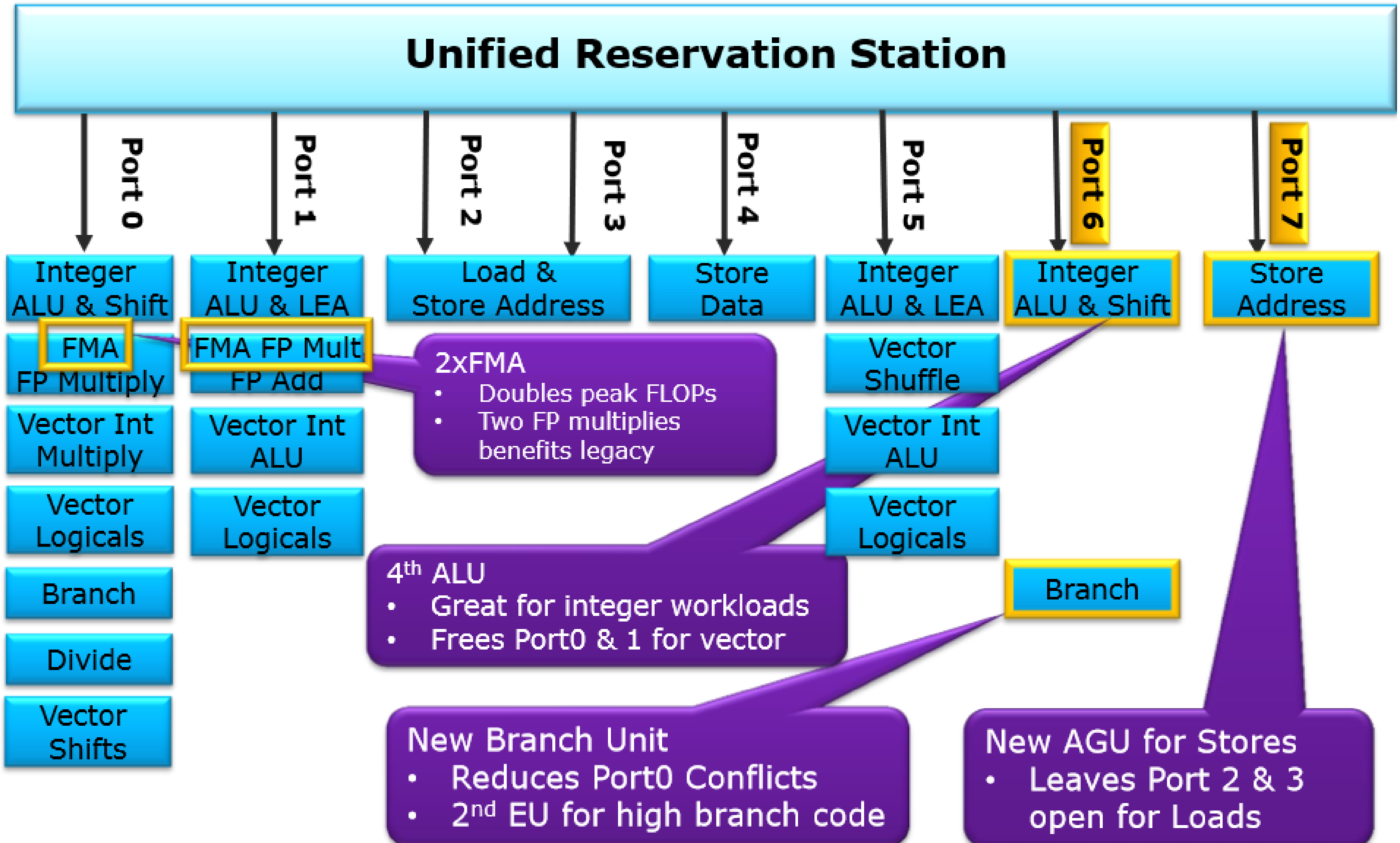
**~1 ns/op**

**Really???**

**Less than 1ns per operation?**

# *Instruction Level Parallelism*

# Haswell Execution Unit Overview



# Access Pattern Benchmark

```
Benchmark                   Score     Error   Units
===================================================
sequential                  0.832  ± 0.006   ns/op
randomPage                  2.703  ± 0.025   ns/op
```

# Access Pattern Benchmark

```
Benchmark                  Score     Error  Units
===============================================
sequential                 0.832  ± 0.006  ns/op
randomPage                 2.703  ± 0.025  ns/op
dependentRandomPage        7.102  ± 0.326  ns/op
```

# Access Pattern Benchmark

```
Benchmark                  Score    Error   Units
=================================================
sequential                 0.832  ± 0.006   ns/op
randomPage                 2.703  ± 0.025   ns/op
dependentRandomPage        7.102  ± 0.326   ns/op
randomHeap                19.896  ± 3.110   ns/op
```

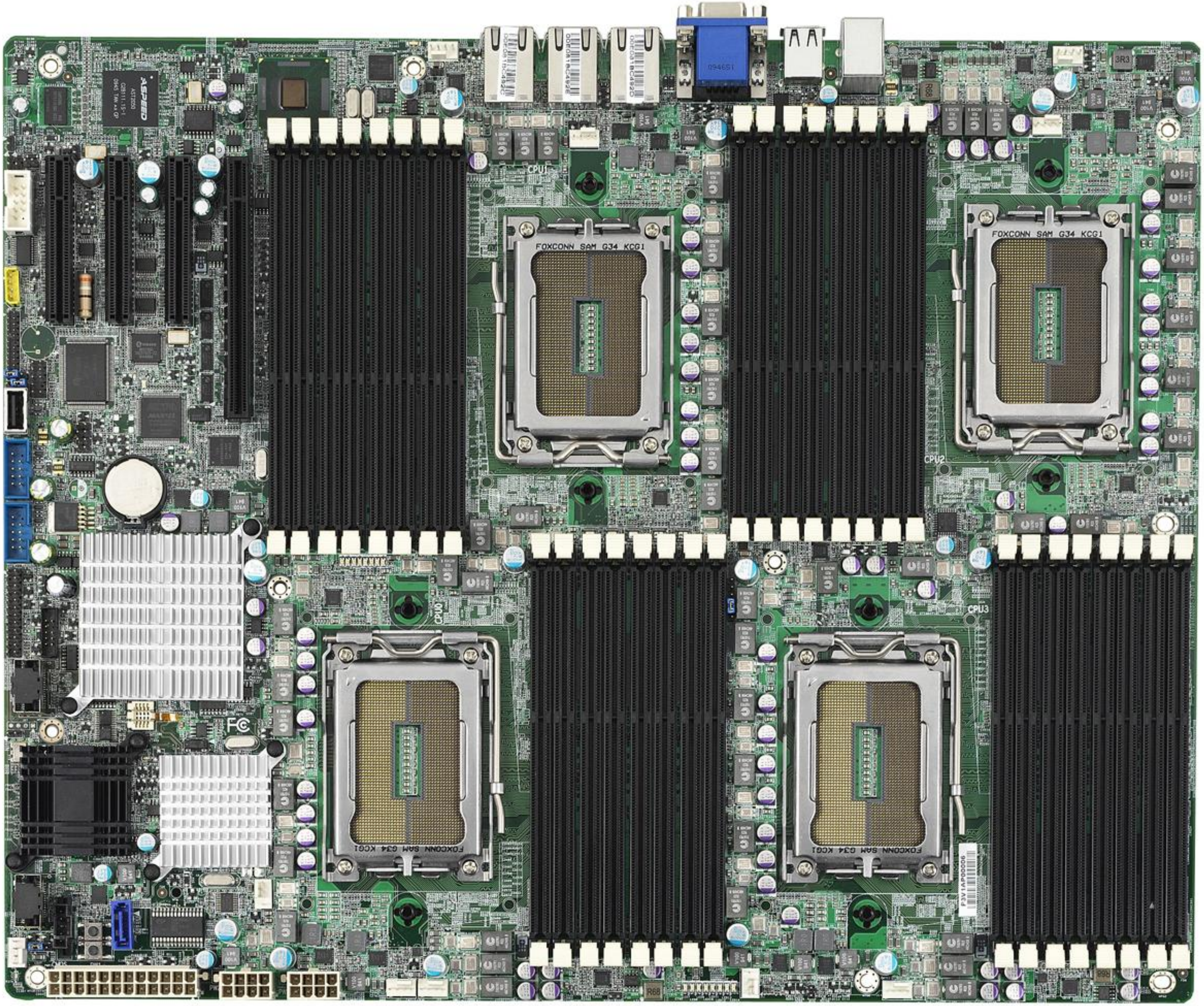
# Access Pattern Benchmark

```
Benchmark                     Score     Error   Units
============================================
sequential                    0.832  ± 0.006   ns/op
randomPage                    2.703  ± 0.025   ns/op
dependentRandomPage           7.102  ± 0.326   ns/op
randomHeap                   19.896  ± 3.110   ns/op
dependentRandomHeap          89.516  ± 4.573   ns/op
```

# Access Pattern Benchmark

```
Benchmark                 Score     Error   Units
=================================================
sequential                0.832  ± 0.006   ns/op
randomPage                2.703  ± 0.025   ns/op
dependentRandomPage       7.102  ± 0.326   ns/op
randomHeap               19.896  ± 3.110   ns/op
dependentRandomHeap      89.516  ± 4.573   ns/op
```

**~90 ns/op**


ASPEED
CPU1
FOXCONN SAM G34 KCG1
FOXCONN SAM G34 KCG1
CPU2
CPU0
CPU3
FOXCONN SAM G34 KCG1
FOXCONN SAM G34 KCG1

A 100ns cache-miss is a lost opportunity to execute ~1000 instructions on CPU

# Algorithms & Data Structures

# Little's Law

$$L = \lambda W$$

<u>Bandwidth Delay Product</u>:

Bytes in flight = Bandwidth * Latency

80 bytes / 100ns = 800 MB/s :10 LFBs

# Little's Law

$$L = \lambda W$$

`Bandwidth Delay Product:`

`Bytes in flight = Bandwidth * Latency`

`80 bytes / 100ns = 800 MB/s :10 LFBs`

`80 bytes / 15ns = 5.3 GB/s :prefectch`

# Little's Law

$$L = \lambda W$$

Bandwidth Delay Product:

Bytes in flight = Bandwidth * Latency

80 bytes / 100ns = 800 MB/s :10 LFBs

80 bytes / 15ns = 5.3 GB/s :prefectch

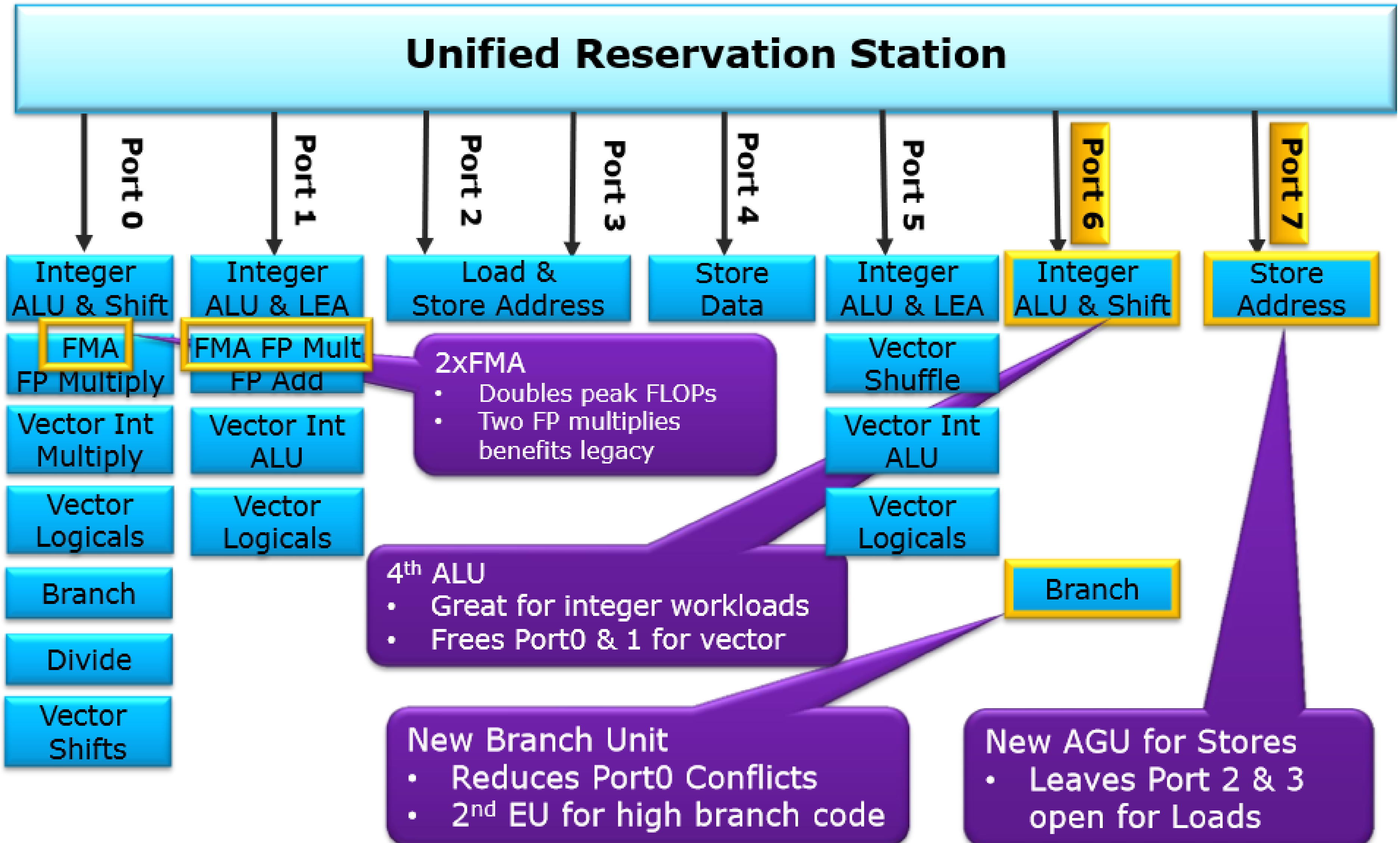
640 bytes / 15ns = 42.6 GB/s :cachelines

***Arrays are the most efficient data structure to traverse***

***Functional data structures are like sausages, the more you see them being made, the less well you will sleep***

## *Branches*

# Haswell Execution Unit Overview



# Branch Benchmark

```
Benchmark                    Score        Error   Units
=========================================================
baseline                   585.600  ±     4.469   us/op
```

# Branch Benchmark

```
Benchmark                Score       Error   Units
==================================================
baseline               585.600   ±   4.469   us/op
predictable            578.364   ±  10.906   us/op
```

# Branch Benchmark

```
Benchmark              Score        Error  Units
===============================================
baseline             585.600 ±      4.469  us/op
predictable          578.364 ±     10.906  us/op
unPredictable       2234.414 ±    564.472  us/op
```

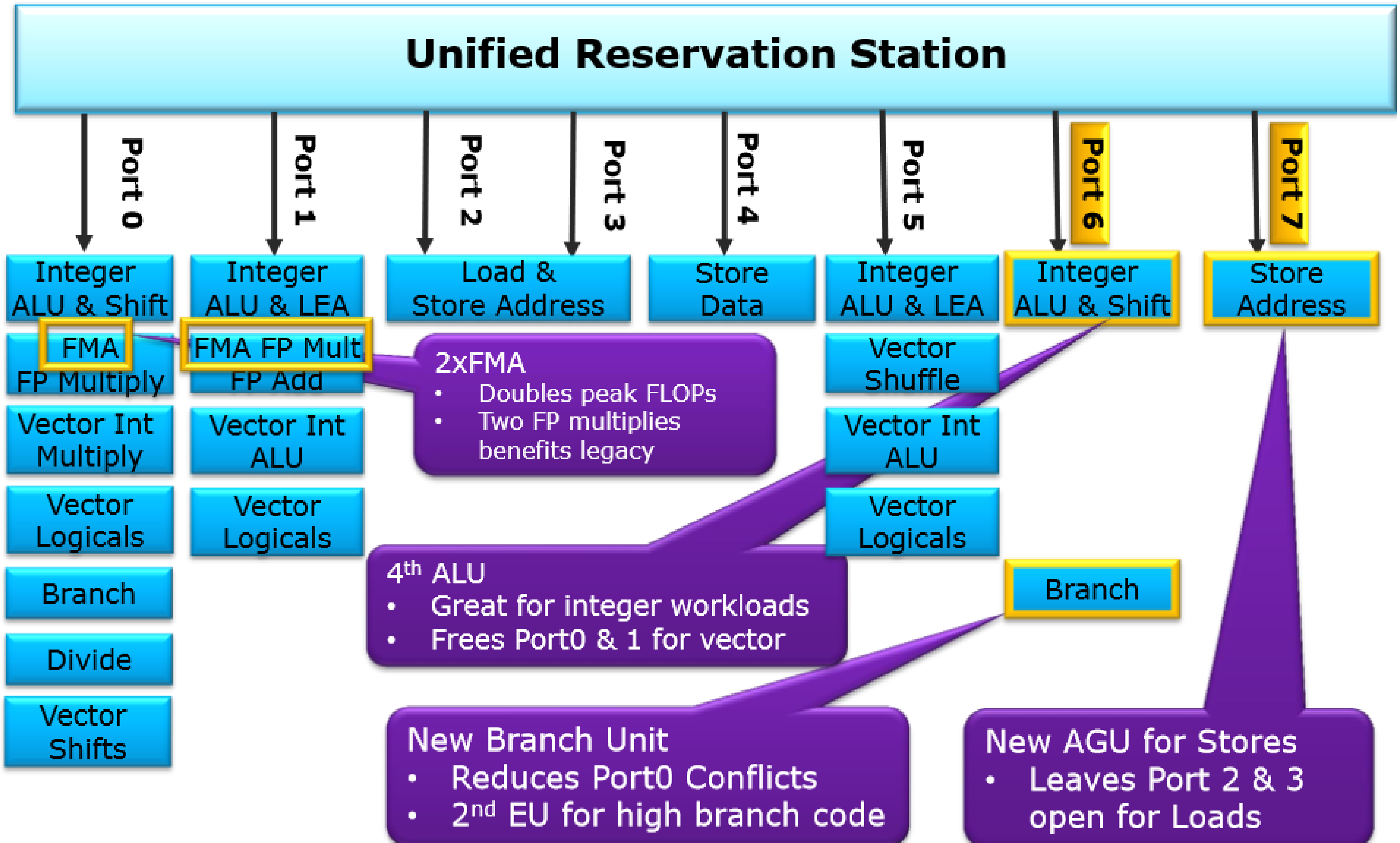
# *What can we do?*

***Count bits as Booleans***

# *Wide Registers*

# *Math, Data Dependencies, and Instruction Level Parallelism*

# Haswell Execution Unit Overview



# *Consider Sorting Arrays*

https://lamport.azurewebsites.net/pubs/multiple-byte.pdf

Programming Techniques

G. Manacher Editor

# Multiple Byte Processing with Full-Word Instructions

Leslie Lamport
Massachusetts Computer Associates, Inc.

**A method is described which allows parallel processing of packed data items using only ordinary full-word computer instructions, even though the processing requires operations whose execution is contingent upon the value of a datum. It provides a useful technique for processing small data items such as alphanumeric characters.**

**Key Words and Phrases: byte processing, character processing, packed data**

**CR Categories: 4.9**

***"It's a neat hack, and it's more useful now than it was then for two reasons."***

***- Leslie Lamport (2011)***

***"The obvious reason is that word size is larger now, with many computers having 64-bit words."***

***- Leslie Lamport (2011)***

*“The less obvious reason is that conditional operations are implemented with masking rather than branching.”*

*- Leslie Lamport (2011)*

*“Branching is more costly on modern multi-issue computers than it was on the computers of the 70s.”*

*- Leslie Lamport (2011)*

https://www.inf.ed.ac.uk/teaching/courses/exc/reading/morris.pdf

Programming Techniques

S.L. Graham, R.L. Rivest
Editors

# Counting Large Numbers of Events in Small Registers

Robert Morris
Bell Laboratories, Murray Hill, N.J.

**It is possible to use a small counter to keep approximate counts of large numbers. The resulting expected error can be rather precisely controlled. An example is given in which 8-bit counters (bytes) are used to keep track of as many as 130,000 events with a relative error which is substantially independent of the number $n$ of events. This relative error can be expected to be 24 percent or less 95 percent of the time (i.e. $\sigma = n/8$). The techniques could be used to advantage in multichannel counting hardware or software used for the monitoring of experiments or processes.**

**Key Words and Phrases: counting**
**CR Categories: 5.11**

Communications of the ACM

October 1978
Volume 21
Number 10

***Work with your CPU caches***

# Memory Access Considerations

1. **Temporal:** group accesses in time

# Memory Access Considerations

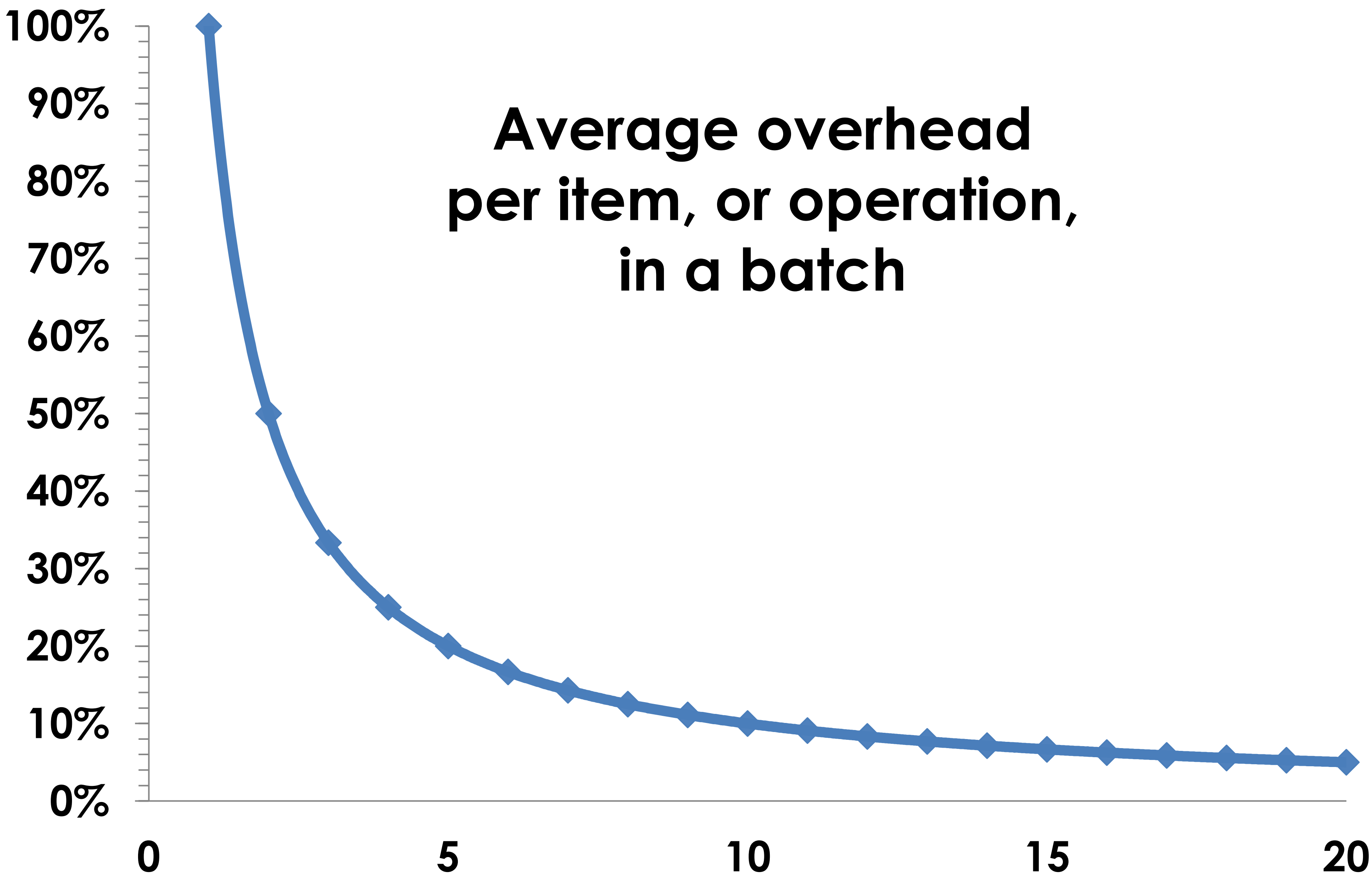
1. **Temporal:** group accesses in time
2. **Spatial:** group access in space

# Memory Access Considerations

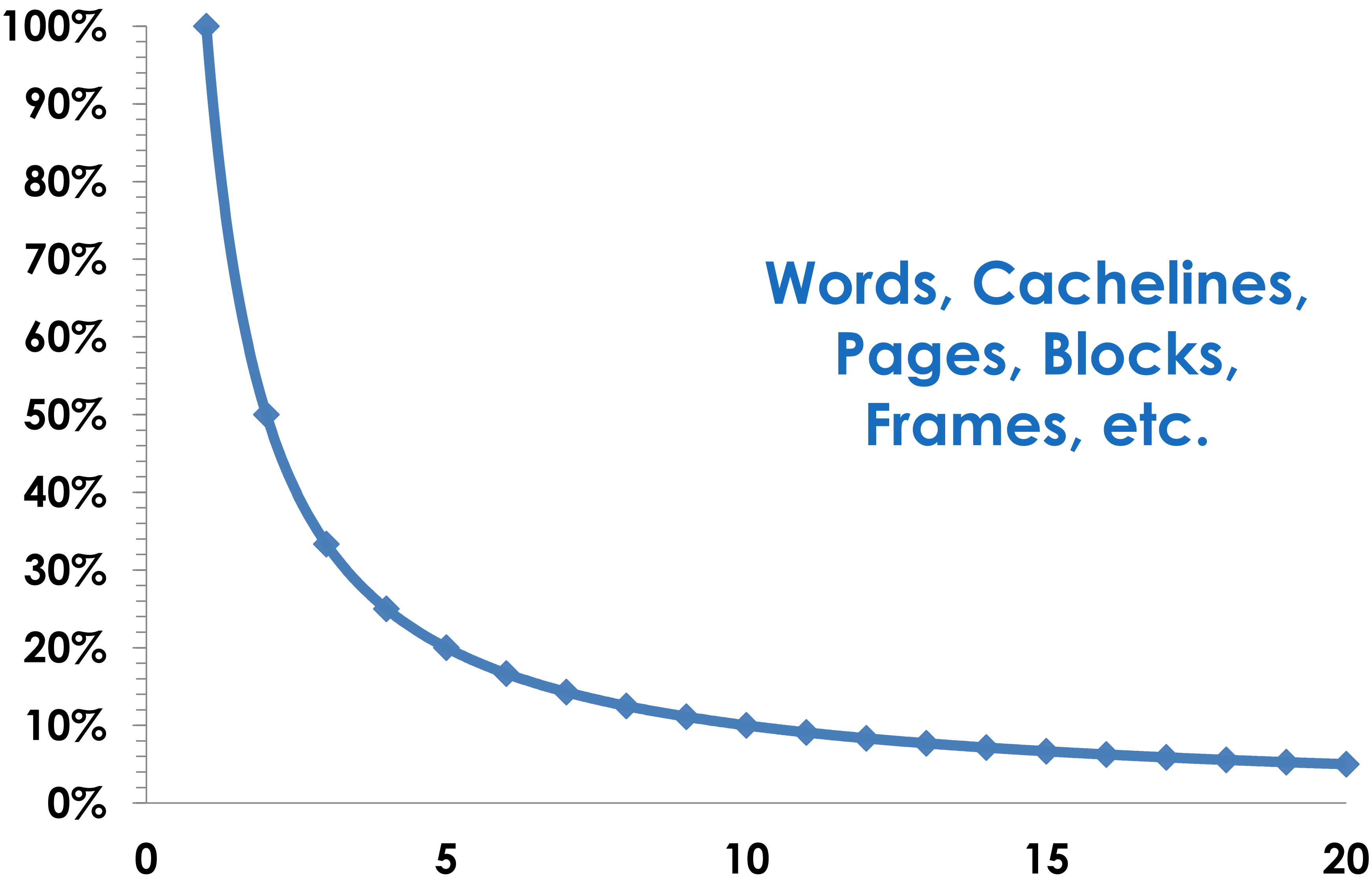
1. **Temporal:** group accesses in time
2. **Spatial:** group access in space
3. **Pattern:** create predictable patterns

# *Batching*

# Batching – Amortising Costs



# Batching – Amortising Costs



***In closing…***

***Profile, profile, profile…***

***Eliminate Waste***
***Batch to Amortise***
***Access Memory in Patterns***
***Favour Math over Branches***
***Favour Predictable Branches***

***Consider Parallelism***

***-***

***ILP & Task***

Is it really “Turtles all the way down”?

# Rectangles all the way down…

Is it really “Turtles all the way down”?

- Networks: Frames
- Operating Systems: Pages
- File systems and storage: Blocks
- DRAM memory: Banks and Row Buffers
- CPU cache subsystems: Cache Lines
- Applications use Arrays plus and interesting data structures are made up of small Arrays

***"I don't care what data structure you use, nothing beats an array"***

***- a HFT Programmer***

# Questions?

**Twitter: @mjpt777**

***"Travel is fatal to prejudice, bigotry, and narrow-mindedness, and many of our people need it sorely on these accounts. Broad, wholesome, charitable views of men and things cannot be acquired by vegetating in one little corner of the earth all one's lifetime."***

*- Mark Twain*