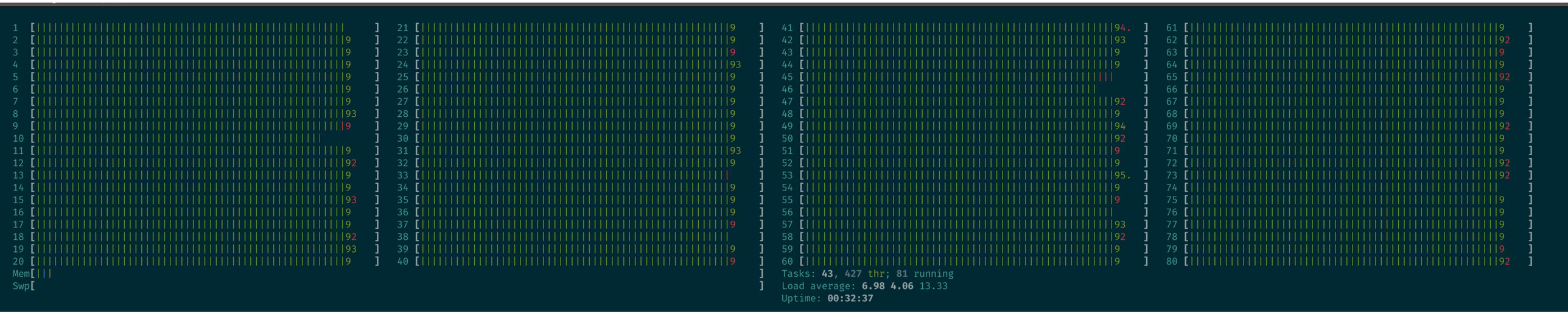
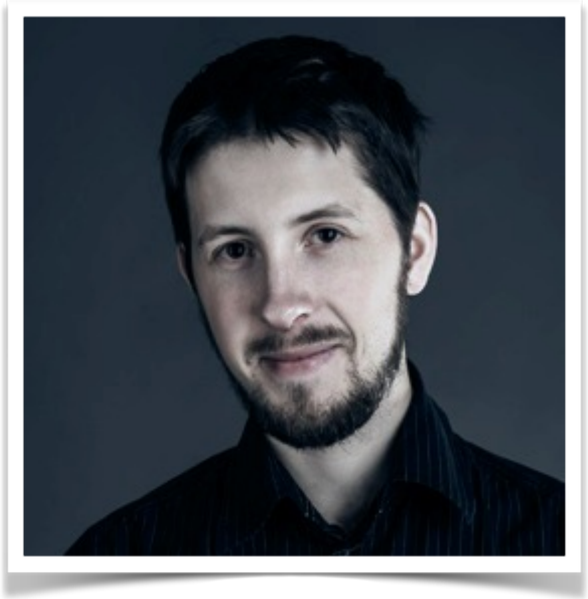
What do I do…
with a 1000 cores

ags,  2k19

JPoint


1 [ ]
2 [9]
3 [9]
4 [9]
5 [9]
6 [9]
7 [9]
8 [93]
9 [9]
10 [ ]
11 [9]
12 [92]
13 [9]
14 [9]
15 [93]
16 [9]
17 [9]
18 [92]
19 [93]
20 [9]
21 [9]
22 [9]
23 [9]
24 [93]
25 [9]
26 [9]
27 [9]
28 [9]
29 [9]
30 [9]
31 [93]
32 [9]
33 [ ]
34 [9]
35 [9]
36 [9]
37 [9]
38 [ ]
39 [9]
40 [9]
41 [94.]
42 [93]
43 [9]
44 [9]
45 [ ]
46 [ ]
47 [92]
48 [9]
49 [94]
50 [92]
51 [9]
52 [9]
53 [95.]
54 [9]
55 [9]
56 [ ]
57 [93]
58 [92]
59 [9]
60 [9]
61 [9]
62 [92]
63 [9]
64 [9]
65 [92]
66 [9]
67 [9]
68 [9]
69 [92]
70 [9]
71 [9]
72 [92]
73 [92]
74 [ ]
75 [9]
76 [9]
77 [9]
78 [9]
79 [9]
80 [92]
Mem[||
Swp[
Tasks: 43, 427 thr; 81 running
Load average: 6.98 4.06 13.33
Uptime: 00:32:37

andrzej grzesik



twitter://ags313

andrzej.grzesik@gmail.com

andrzejgrzesik.info

about:me

polish java user group

GeeCON

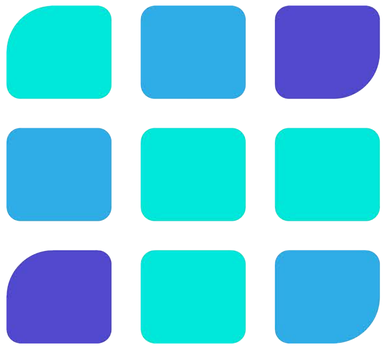
[sckrk]

cracow hadoop group


lives
from
JPoint
Faroe Islands
Sweden
Finland
Norway
Gulf of Bothnia
Oslo
Stockholm
Helsinki
St Petersburg
Санкт-Петербург
Tallinn
Estonia
Baltic Sea
Moscow
Москва
Nizhny Novgorod
Нижний Новгород
Kazan
Казань
Izhevsk
Ижевск
Perm
Пермь
Ufa
Уфа
Samara
Самара
Saratov
Саратов
Riga
Latvia
Lithuania
Vilnius
Minsk
Мінск
Belarus
Voronezh
Воронеж
Volgograd
Волгоград
Rostov-on-Don
Ростов-на-Дону
Krasnodar
Краснодар
Edinburgh
United Kingdom
Ireland
Dublin
Liverpool
North Sea
Denmark
Hamburg
Amsterdam
Netherlands
Brussels
Belgium
Luxembourg
Frankfurt
Berlin
Poland
Warsaw
Prague
Czechia
Paris
France
Munich
Vienna
Slovakia
Budapest
Hungary
Austria
Switzerland
Bay of Biscay
Milan
Slovenia
Zagreb
Croatia
Belgrade
Београд
Bosnia and Herzegovina
Serbia
Sarajevo
Monaco
Andorra
Italy
Rome
Montenegro
Podgorica
Подгорица
Kosovo
Macedonia (FYROM)
Tirana
Portugal
Lisbon
Porto
Madrid
Barcelona
Ukraine
Kharkiv
Moldova
Chisinau
Odesa
Одеса
Romania
Bucharest
Black Sea
Sofia
София
Bulgaria
Istanbul
Bursa
Ankara
Turkey
Georgia
Tbilisi
Armenia

100s of machines

since 2010

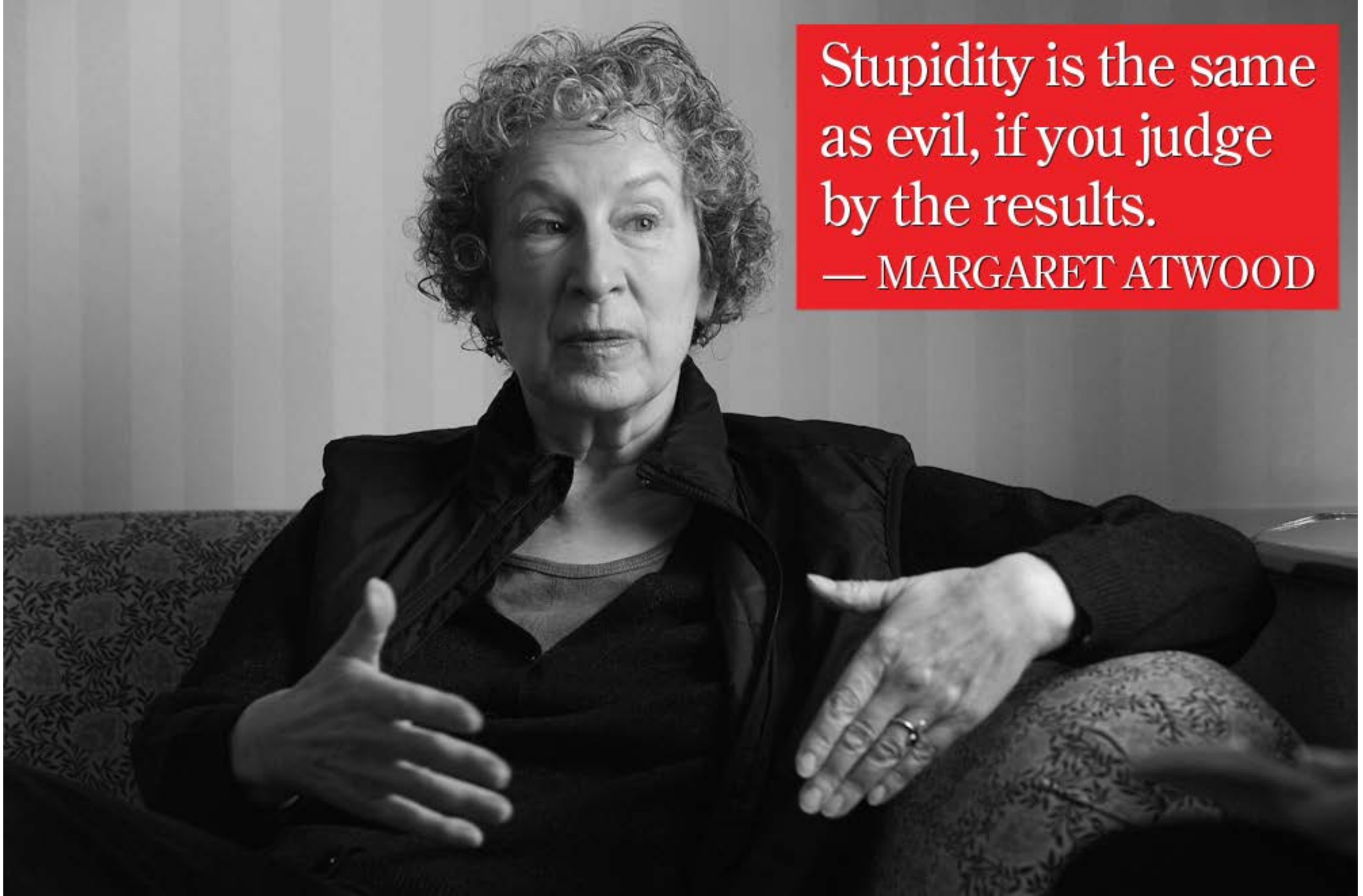


SIMUDYNE

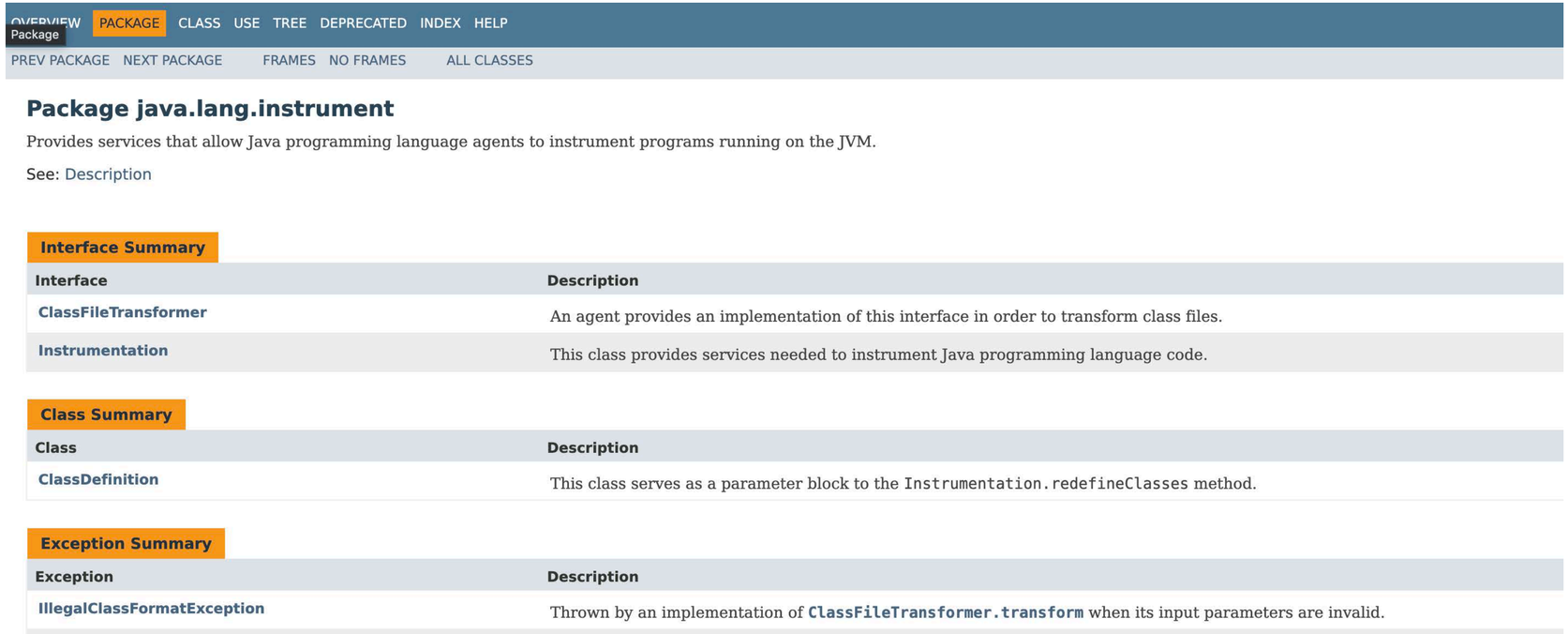
my opinions are my own

questions?

just ask!


Stupidity is the same as evil, if you judge by the results.
— MARGARET ATWOOD

Nothing to do with Java Agents



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Package

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

Package java.lang.instrument

Provides services that allow Java programming language agents to instrument programs running on the JVM.

See: Description

Interface Summary

Interface	Description
ClassFileTransformer	An agent provides an implementation of this interface in order to transform class files.
Instrumentation	This class provides services needed to instrument Java programming language code.

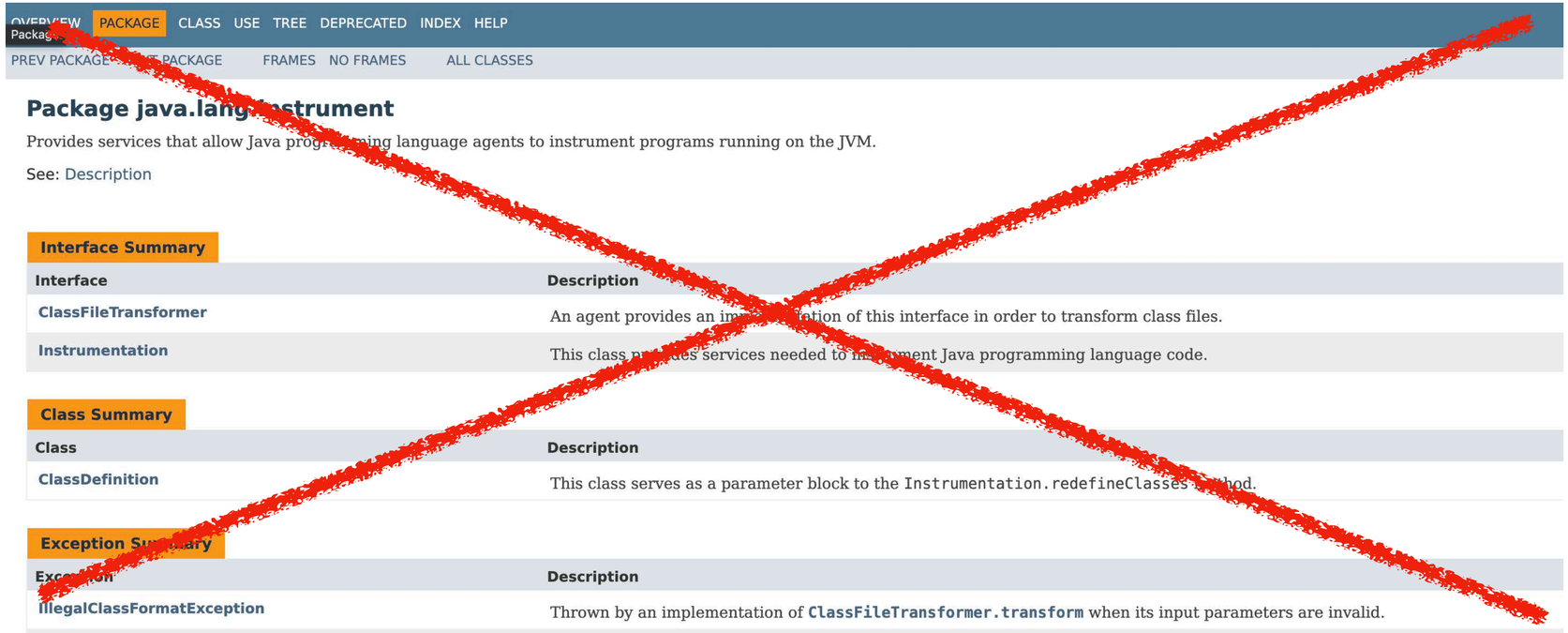
Class Summary

Class	Description
ClassDefinition	This class serves as a parameter block to the `Instrumentation.redefineClasses` method.

Exception Summary

Exception	Description
IllegalClassFormatException	Thrown by an implementation of `ClassFileTransformer.transform` when its input parameters are invalid.

Nothing to do with Java Agents



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

Package java.lang.instrument

Provides services that allow Java programming language agents to instrument programs running on the JVM.

See: Description

Interface Summary

Interface	Description
ClassFileTransformer	An agent provides an implementation of this interface in order to transform class files.
Instrumentation	This class provides services needed to instrument Java programming language code.

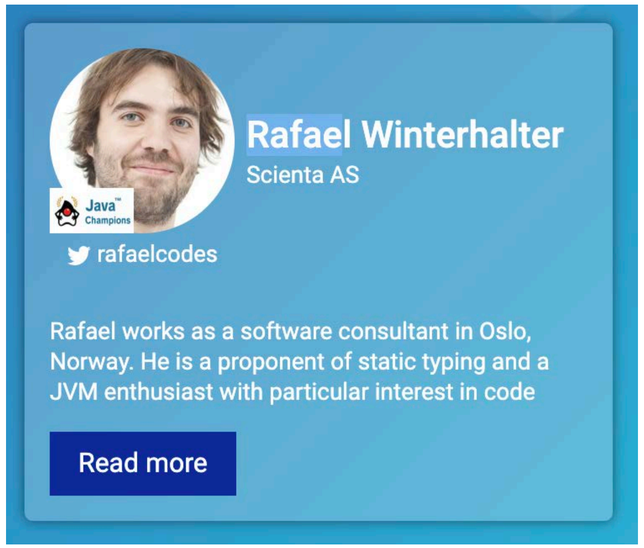
Class Summary

Class	Description
ClassDefinition	This class serves as a parameter block to the `Instrumentation.redefineClasses` method.

Exception Summary

Exception	Description
IllegalClassFormatException	Thrown by an implementation of `ClassFileTransformer.transform` when its input parameters are invalid.

For Java Agents, talk to:



May you live in interesting times

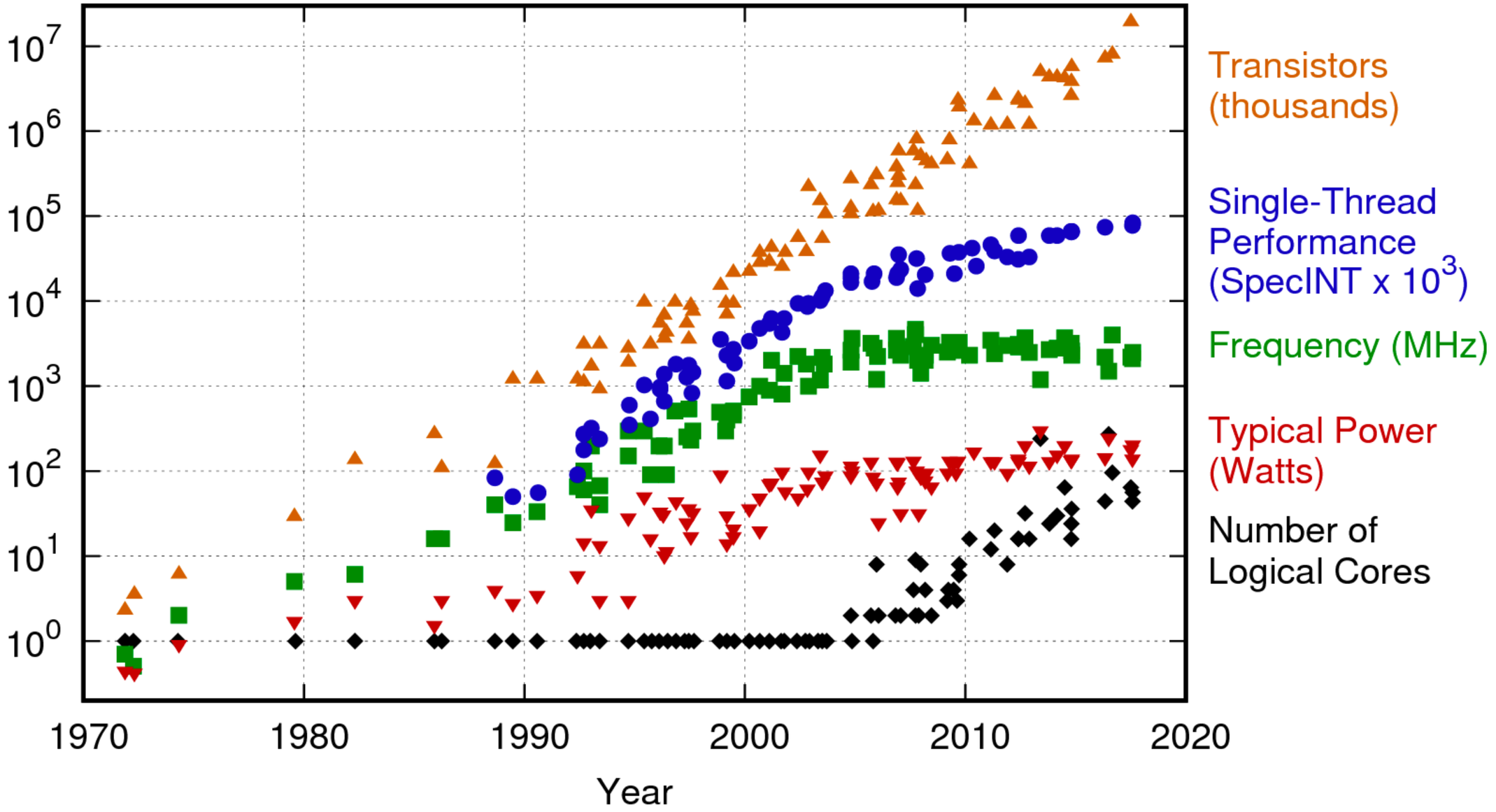
May you live in interesting times

ancient Chinese curse, at least according to Terry Pratchett

Your 1st computer?

Your 1st computer?

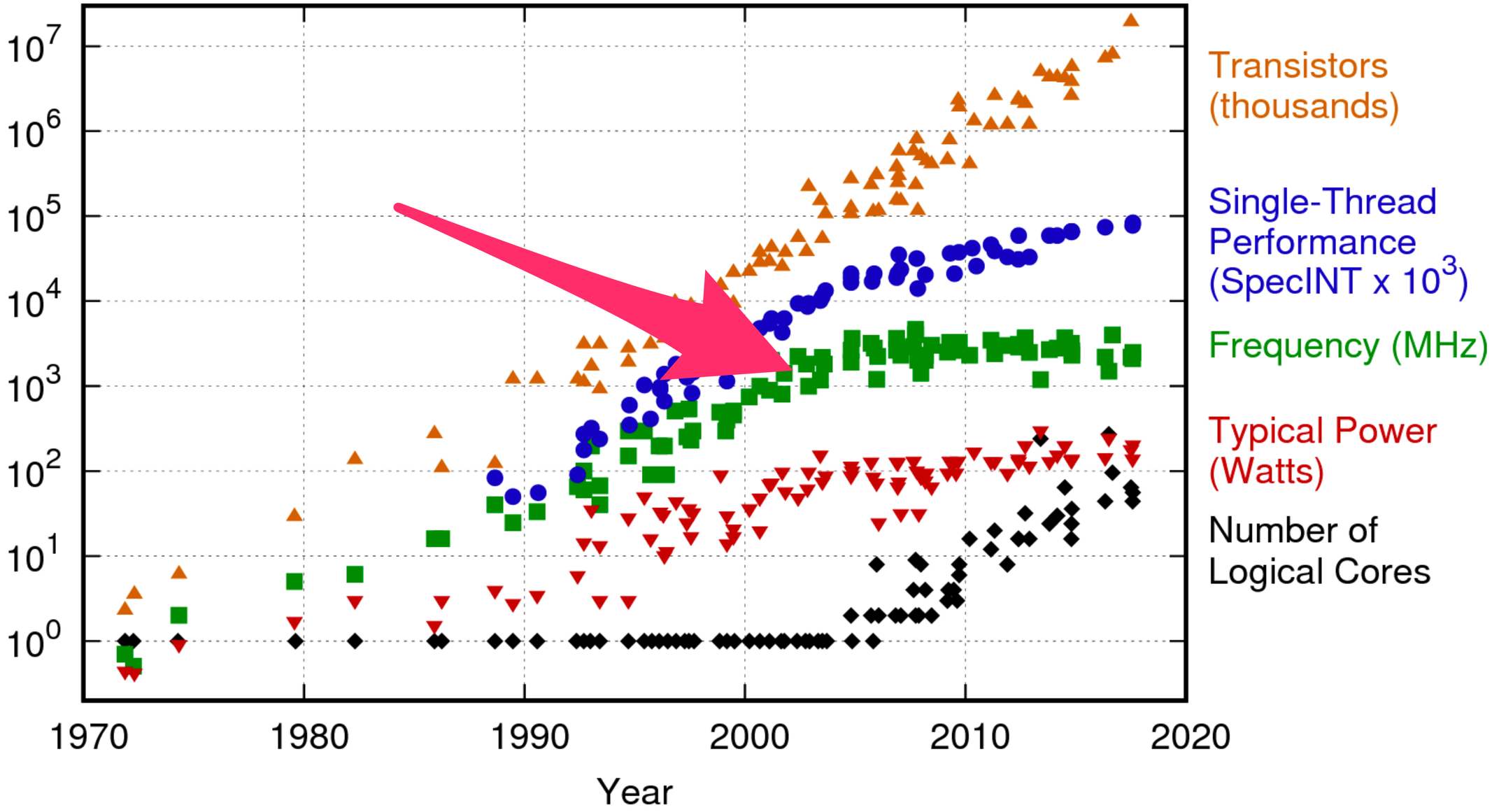
42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://github.com/karlrupp/microprocessor-trend-data

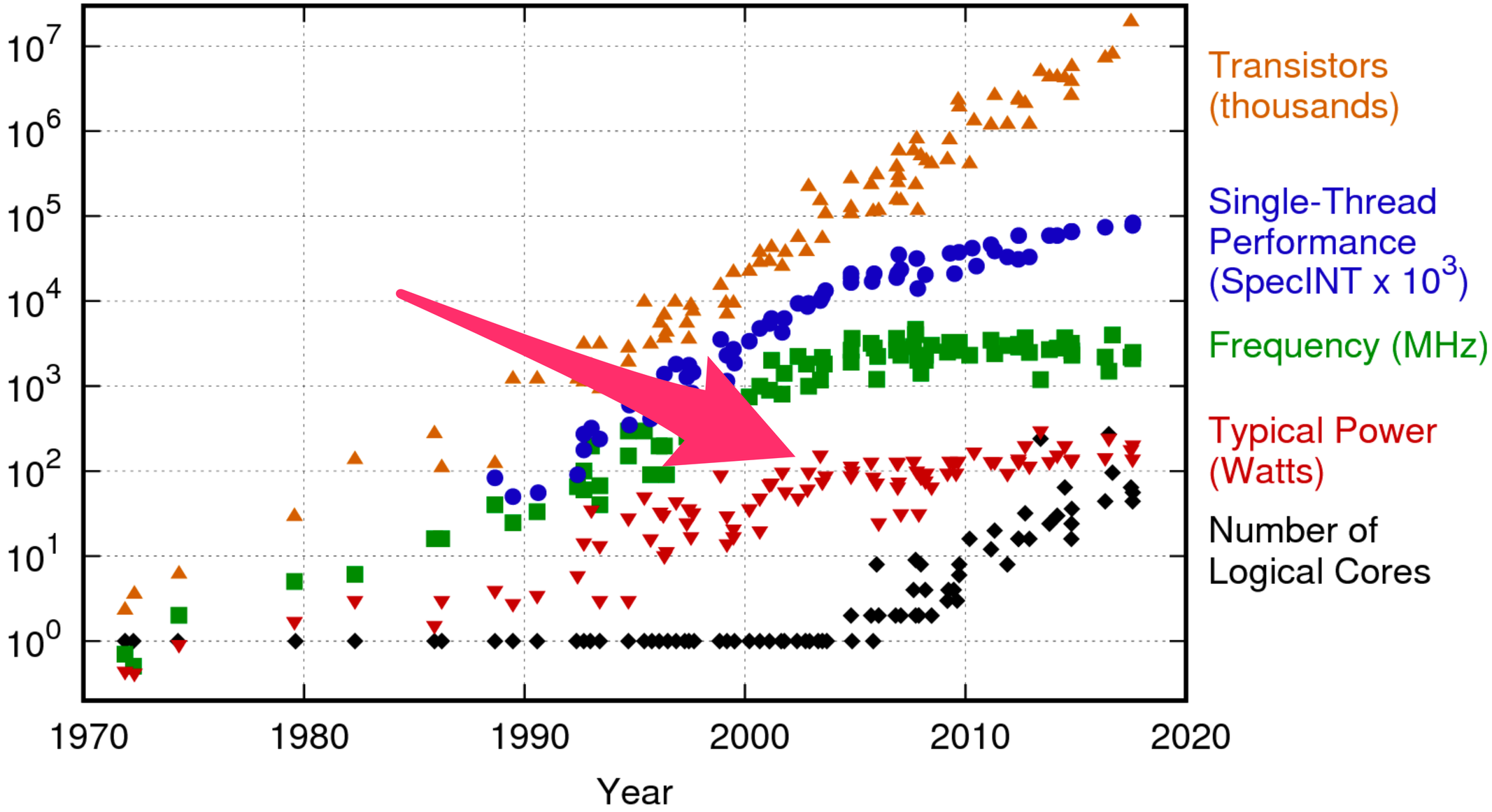
42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

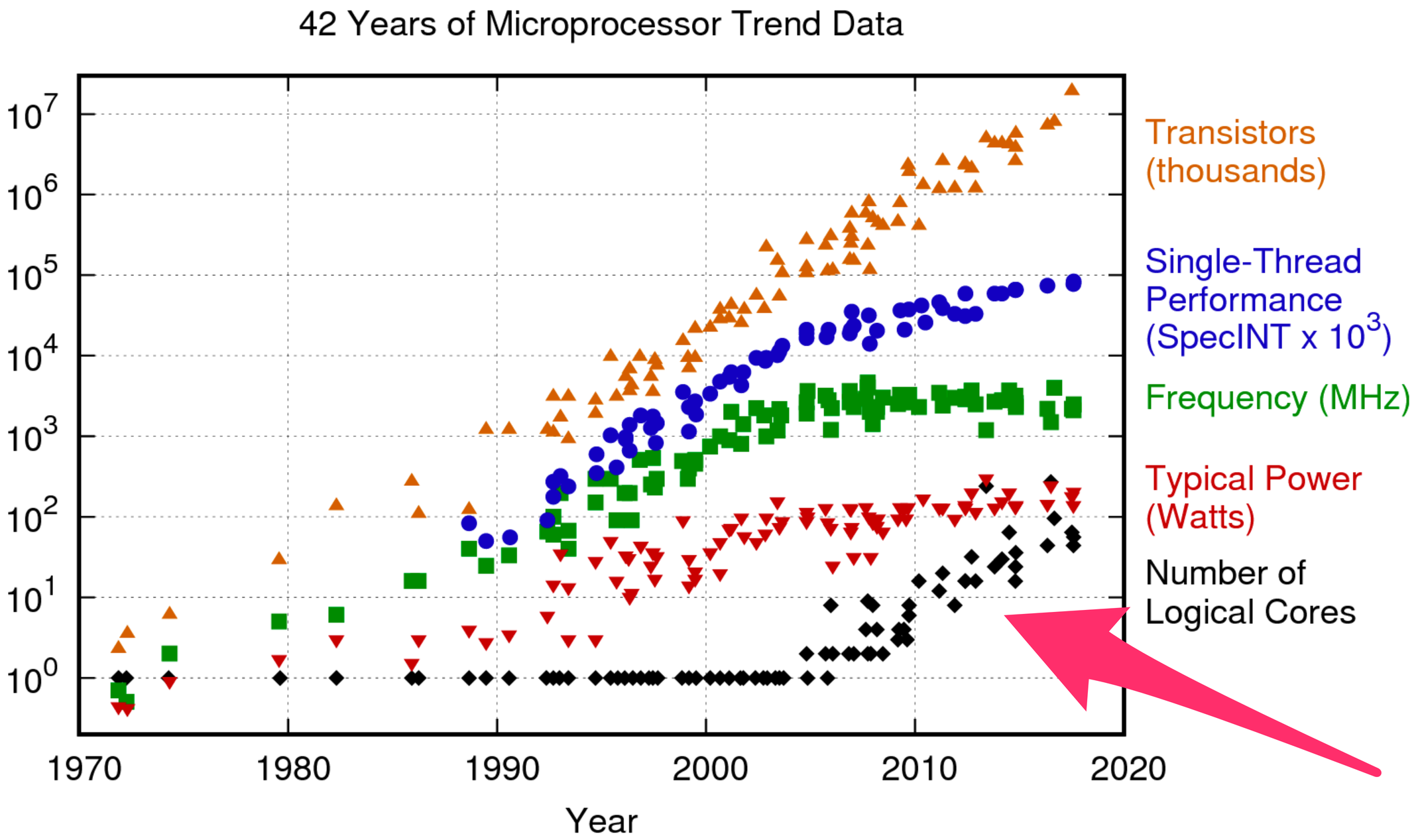
https://github.com/karlrupp/microprocessor-trend-data

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://github.com/karlrupp/microprocessor-trend-data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://github.com/karlrupp/microprocessor-trend-data

Disclaimer

Treat as “work report”, not “recommendations”

TOP 500
The List.

top500, June 2010, position 37

37	Japan Agency for Marine-Earth Science and Technology Japan	**Earth Simulator** - SX-9/E/1280M160 NEC	1,280	122.4	131.1

https://www.top500.org/list/2010/06/?page=1

Amazon

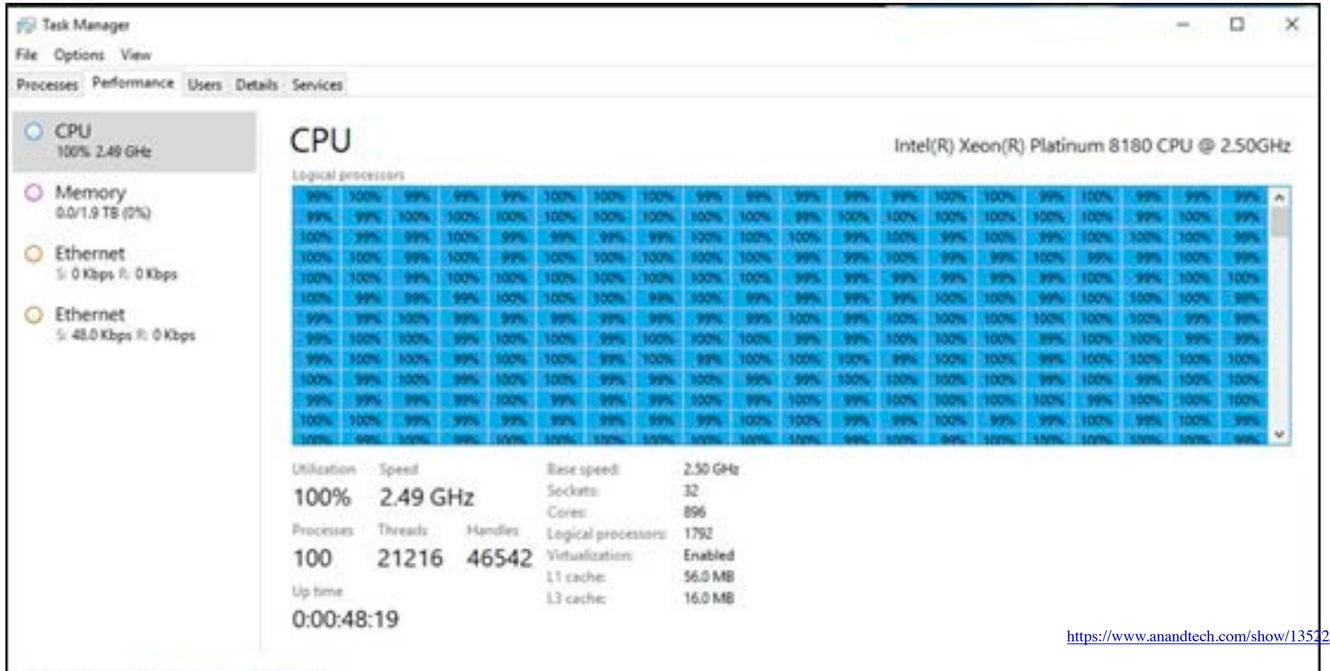
Model	Logical Proc*	Mem (TiB)	Network Perf. (Gbps)	Dedicated EBS Bandwidth (Gbps)
u-6tb1.metal	448	6	25	14
u-9tb1.metal	448	9	25	14
u-12tb1.metal	448	12	25	14

Today

Azure

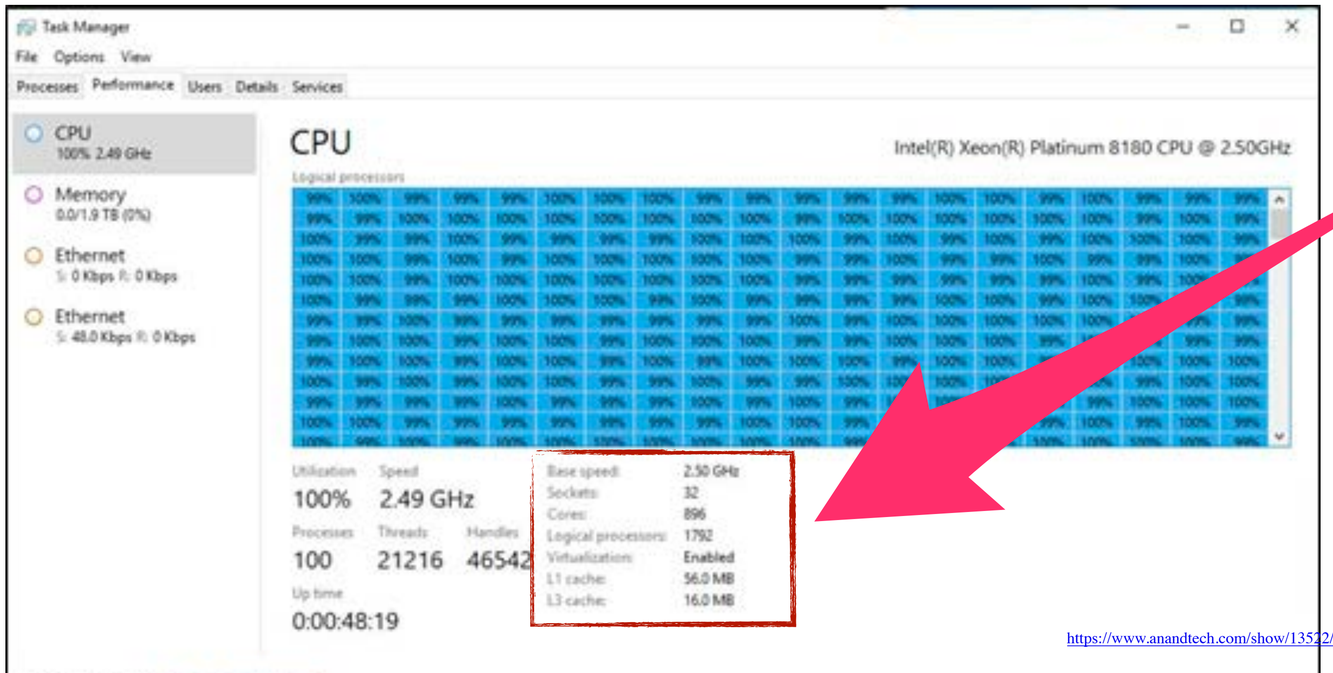
INSTANCE	CORE	RAM	NFS
S96	96	768 GiB	3,072 GiB
S192	192	2,048 GiB	8,192 GiB
S192m	192	4,096 GiB	16,384 GiB
S192xm	192	6,144 GiB	16,384 GiB
S384	384	4,096 GiB	16,384 GiB
S384m	384	6,144 GiB	18,432 GiB
S384xm	384	8,192 GiB	22,528 GiB
S384xxm	384	12,000 GiB	28,250 GiB
S576m	576	12,000 GiB	28,250 GiB

Then Microsoft blogged



https://www.anandtech.com/show/13522/896-xeon-cores-in-one-pc-microsofts-new-x86-datacenter-class-machines-running-windows

Then Microsoft blogged



https://www.anandtech.com/show/13522/896-xeon-cores-in-one-pc-microsofts-new-x86-datacenter-class-machines-running-windows

TOP 10 Sites for November 2018

For more information about the sites and systems in the list, click on the links or view the complete list.

1-100 | 101-200 | 201-300 | 301-400 | 401-500

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

What makes $ sense?

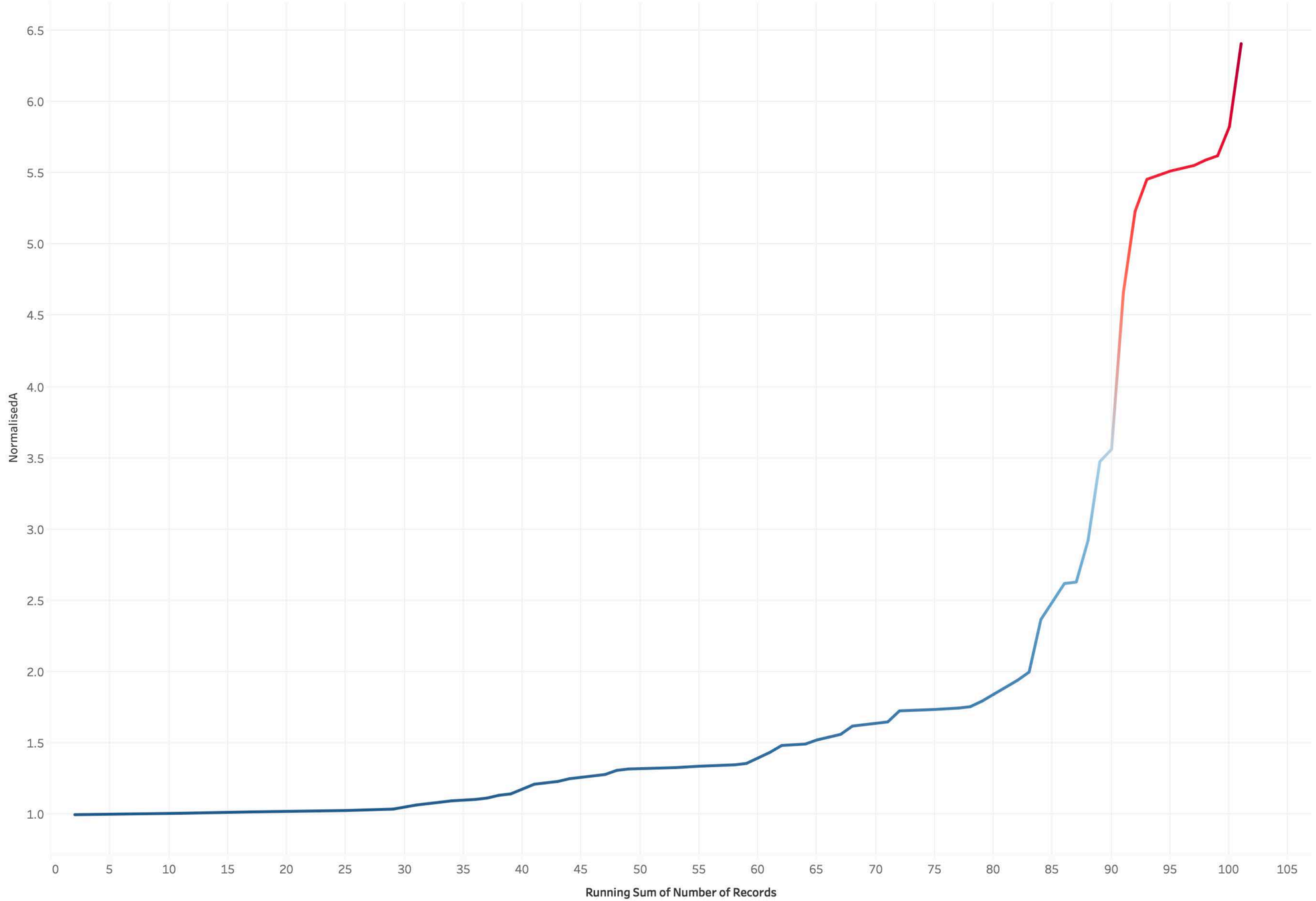
(which one should I buy?)

it depends ;-)

CLOUD	INSTANCE TYPE	NUM	OS	COST/ HR	CPUS	MEM (GB)	NVME SSDS
OCI	VM.Standard2.16	3	Oracle Linux	3.06	96	720	0
OCI	VM.Standard2.16	3	Ubuntu	3.06	96	720	0
GCP	custom-36-73728-discounted	3	Ubuntu	3.18	108	216	0
OCI	BM.Standard2.52	1	Oracle Linux	3.32	104	768	0
OCI	BM.Standard2.52	1	Ubuntu	3.32	104	768	0
Azure	Standard_F32s_v2	3	Ubuntu	4.05	96	192	0
GCP	custom-36-73728	3	Ubuntu	4.54	108	216	0
AWS	c5.9xlarge	3	Ubuntu	4.59	108	216	0
AWS	c5.9xlarge	3	Amazon Linux	4.59	108	216	0
Azure	Standard_F16	6	Ubuntu	4.80	96	192	0
OCI	BM.HPC2.36	2	Oracle Linux	5.40	144	768	2
OCI	BM.HPC2.36	2	Ubuntu	5.40	144	768	2
OCI	VM.DenseIO2.16	3	Ubuntu	6.12	96	720	6
AWS	r5d.12xlarge	2	Ubuntu	6.91	96	768	4
AWS	r5d.12xlarge	2	Amazon Linux	6.91	96	768	4
Azure	Standard_E64s_v3	2	Ubuntu	7.26	128	864	0

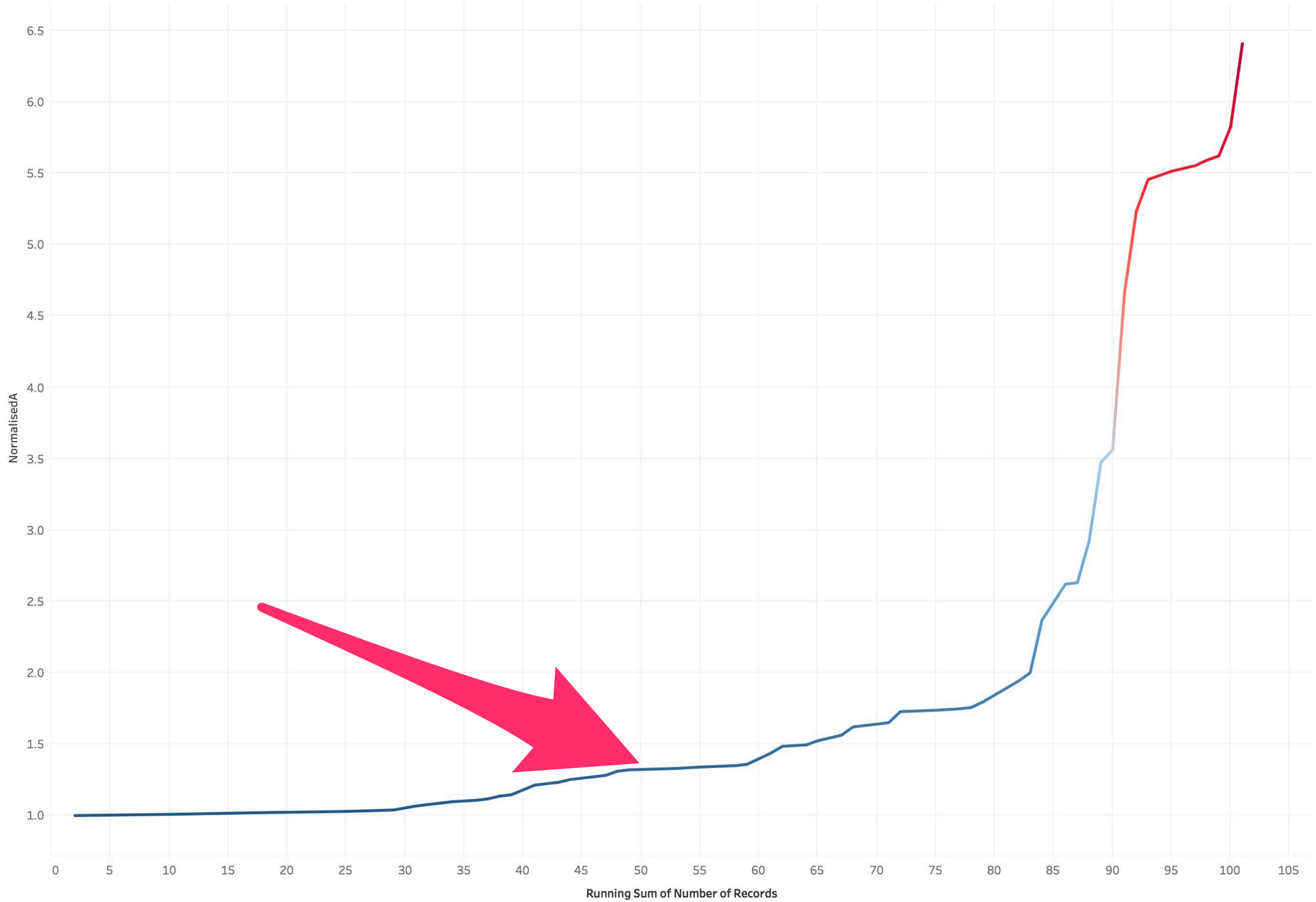
https://www.pilosa.com/blog/cloud-bench-redux/

Workload A



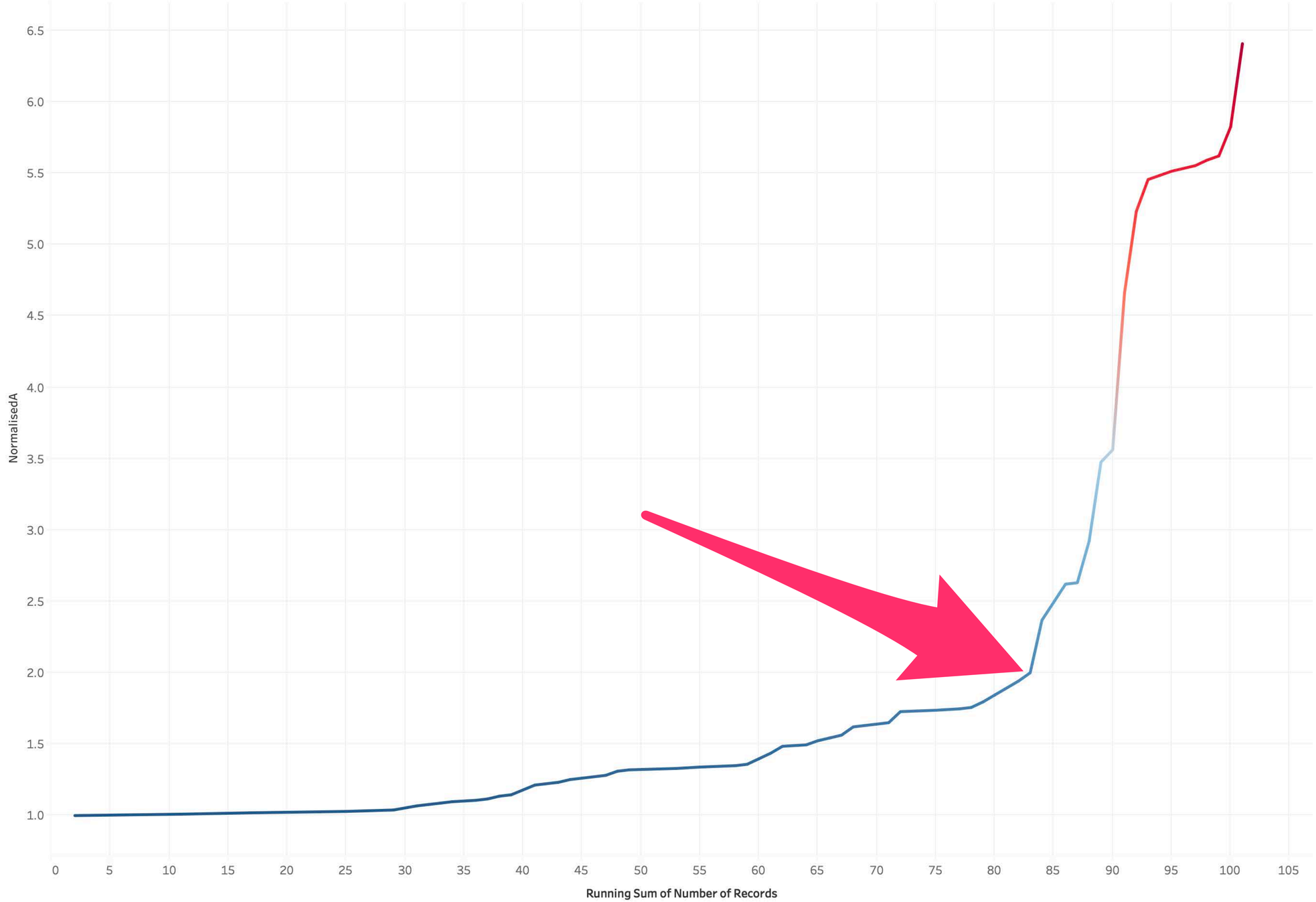
The trend of Running Sum of Number of Records for NormalisedA. Color shows details about NormalisedA.

Workload A



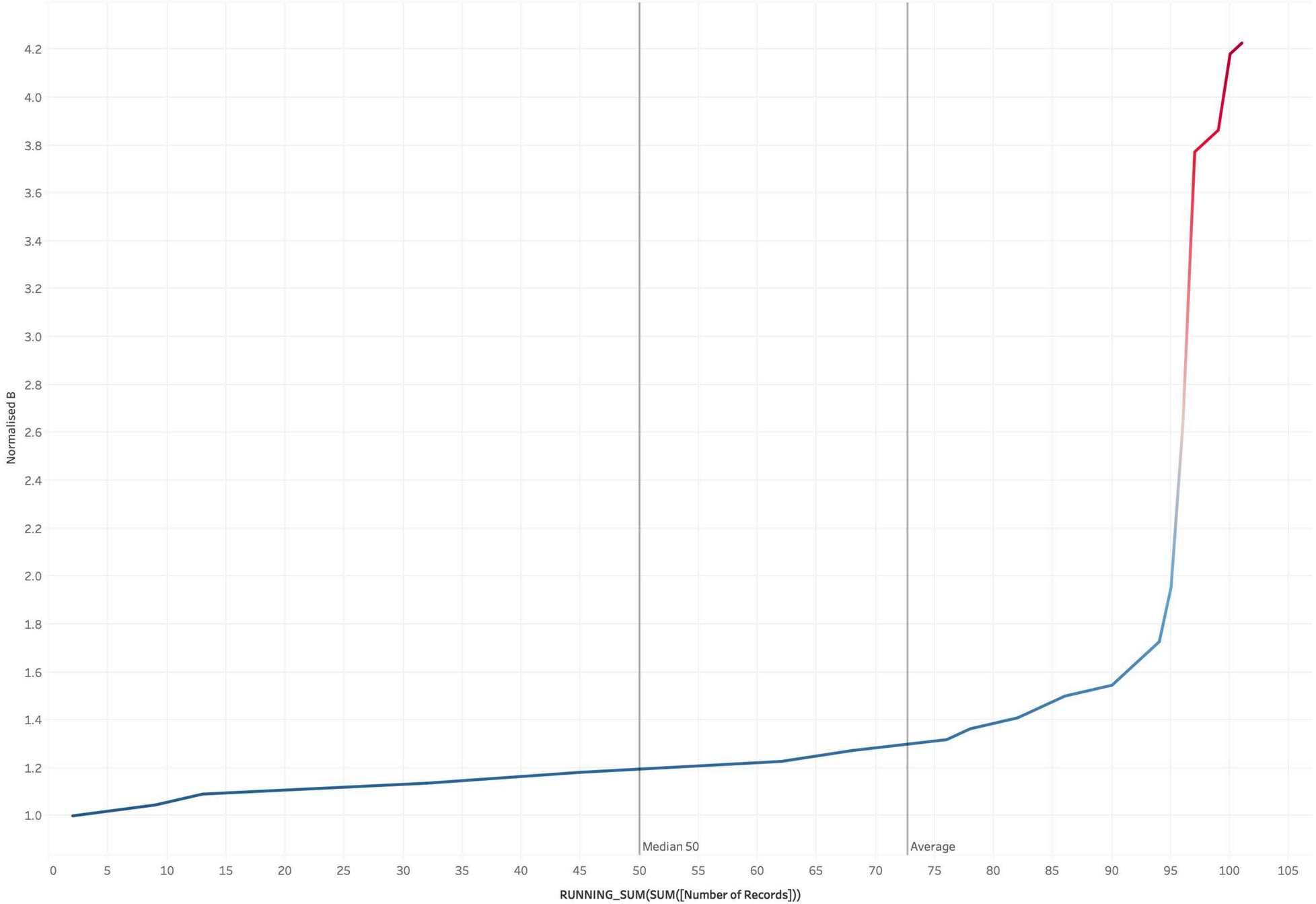
The trend of Running Sum of Number of Records for NormalisedA. Color shows details about NormalisedA.

Workload A

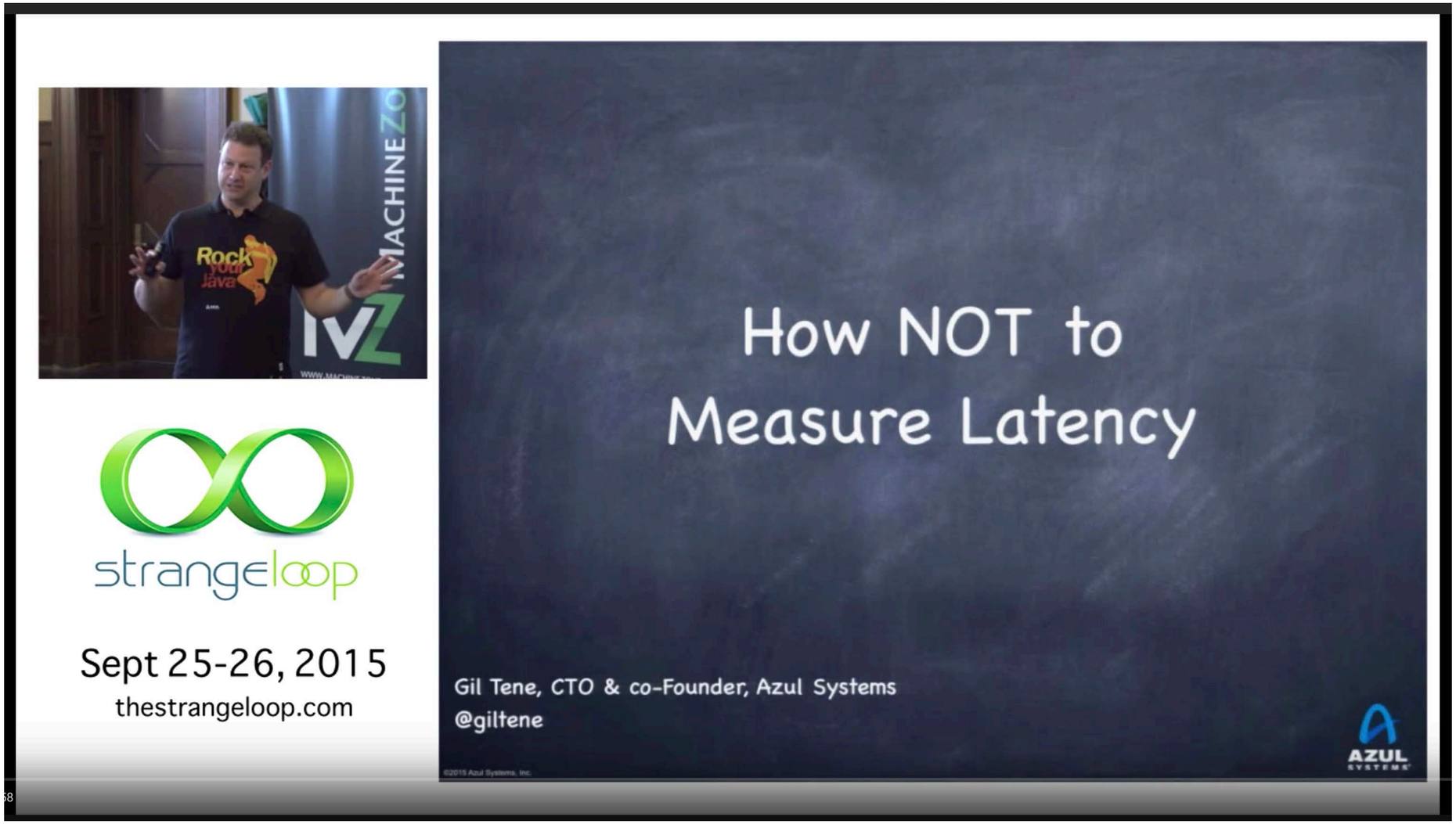


The trend of Running Sum of Number of Records for NormalisedA. Color shows details about NormalisedA.

Workload B



The trend of RUNNING_SUM(SUM([Number of Records])) for Normalised B. Color shows details about Normalised B. Percents are based on the whole table .



https://www.youtube.com/watch?v=lJ8ydIuPFeU

Agent Based modelling

Agent Based Modelling

An **agent-based model** (**ABM**) is a class of computational models for simulating the actions and interactions of autonomous agents (both individual or collective entities such as organizations or groups) with a view to assessing their effects on the system as a whole

https://en.wikipedia.org/wiki/Agent-based_model

agents,
interacting with agents
and environment

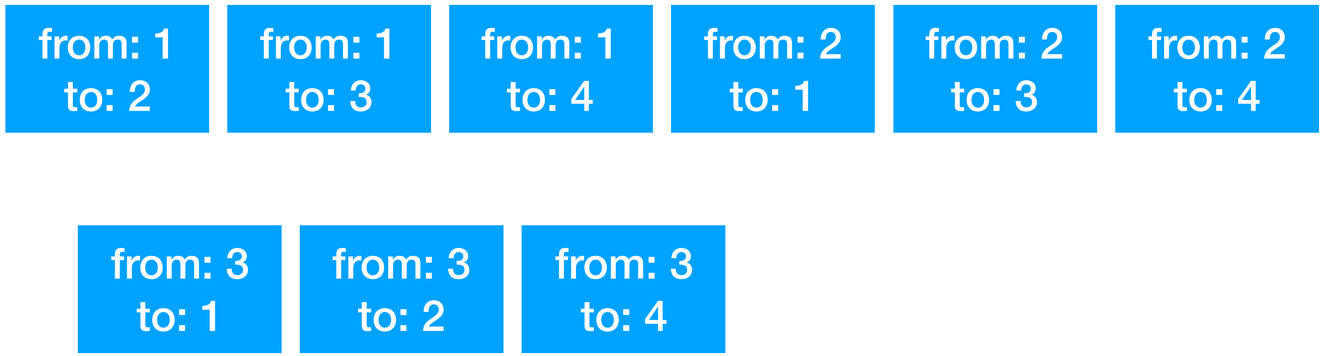
Game of Life

https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

```java
public class Cell extends Agent<GameOfLife.Globals> {
  public static Action<Cell> action(SerializableConsumer<Cell> action) { return Action.create(Cell.class, action); }

  @Variable public boolean alive;

  public void onStart() { getLinks(Links.Neighbour.class).send(Messages.Alive.class, alive); }

  public void onNeighbourMessages() {
    long liveNeightbours = 0;

    List<Messages.Alive> messages = getMessagesOfType(Messages.Alive.class);
    liveNeightbours = // sum

    if (alive && (liveNeightbours < 2 || liveNeightbours > 3)) {
      getLongAccumulator( s: "died").add(1);
      alive = false;
    } else if (!alive && liveNeightbours == 3) {
      getLongAccumulator( s: "born").add(1);
      alive = true;
    }
  }
}
```

https://github.com/simudyne/simudyne-maven-java

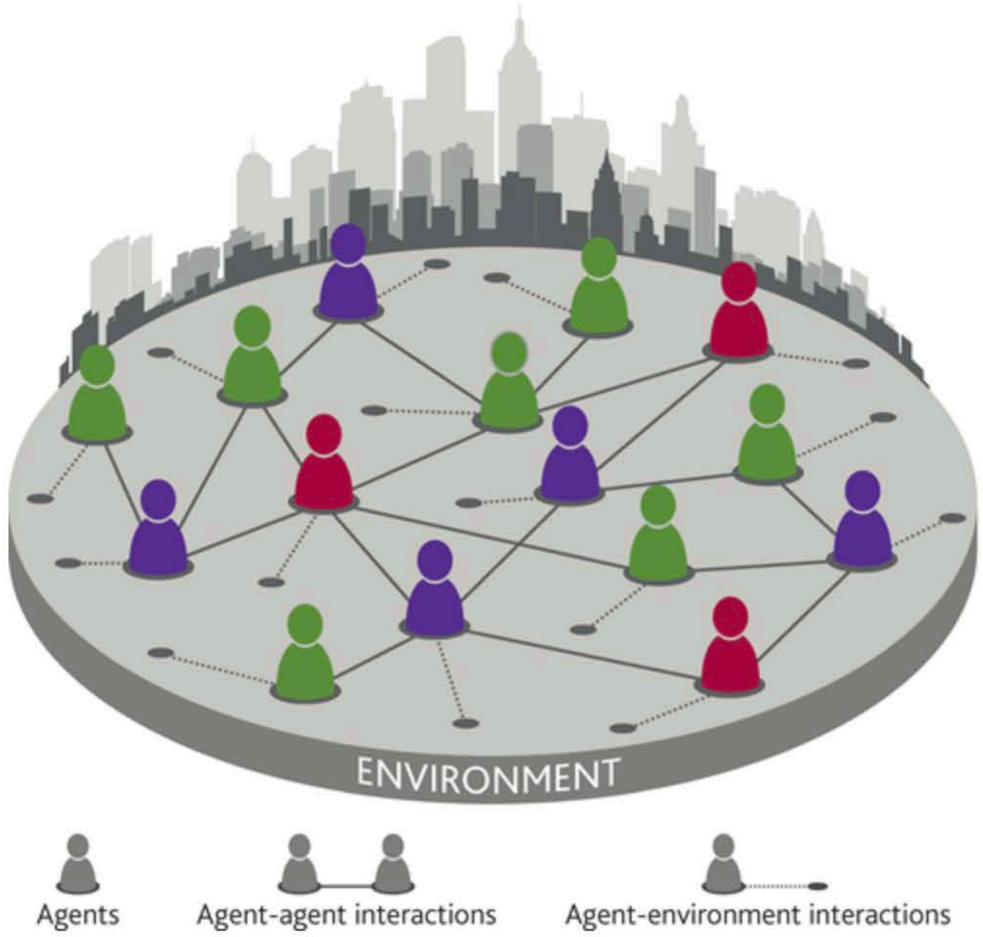
Communications



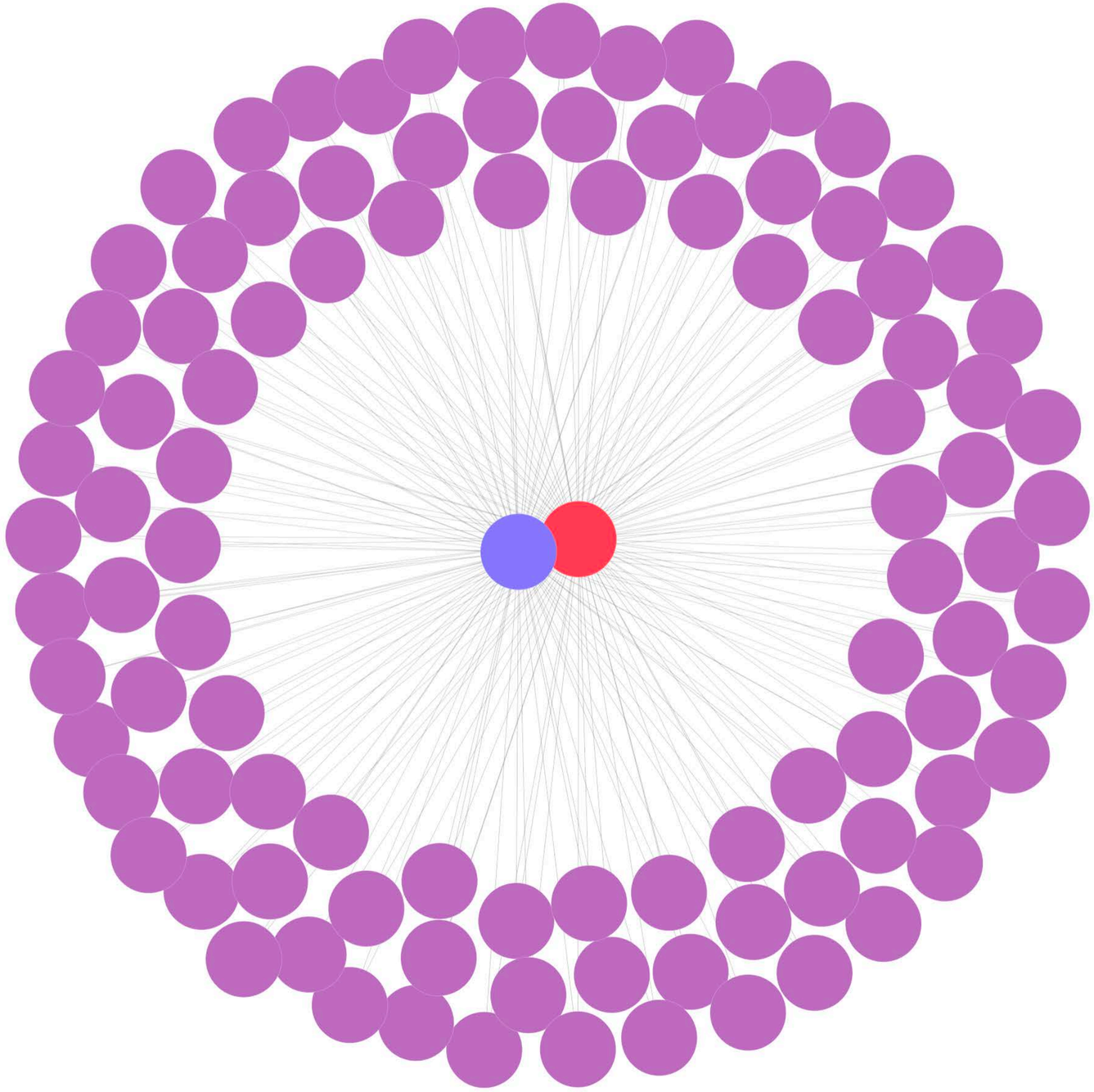
Published on 16 December 2016

By Arthur Turrell

This article considers the strengths of agent-based modelling, which explains the behaviour of a system by simulating the behaviour of each individual 'agent' in it, and the ways that it can be used to help central banks understand the economy.



https://www.bankofengland.co.uk/quarterly-bulletin/2016/q4/agent-based-models-understanding-the-economy-from-the-bottom-up



Implementations?

some implementations:
https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software

Who uses ABMs?

Why Java?

Loved, tested,
popular, compatible,
stable…

Great time to build ABMs

because hardware!

This probably does not apply to a typical * app

insert Spring/Hibernate/… as required

ags313
@ags313

I like working with distributed systems. Each of my personalities has something to do.

10:44 AM - 18 Dec 2018

Requirements

optimising for wall clock time

Challenges

These numbers (might) represent something,
do *not* draw any conclusions.

concurrent counter?

synchronised { … }

concurrent counter?

synchronised { … }

java.util.concurrent.atomic.AtomicLong

concurrent counter?

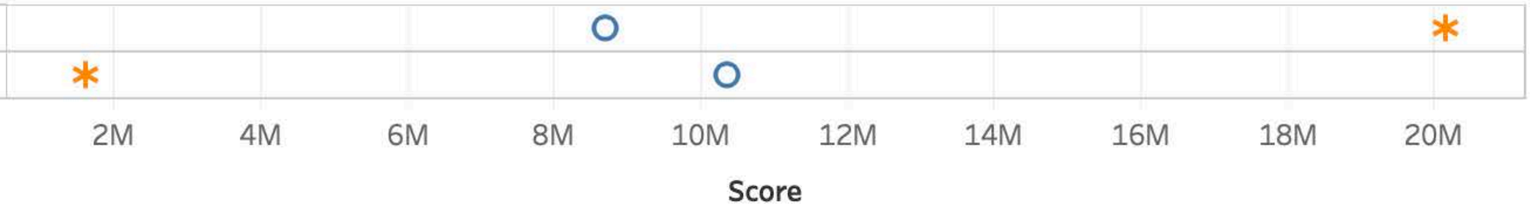
synchronised { … }

java.util.concurrent.atomic.AtomicLong

java.util.concurrent.atomic.LongAdder

AtomicCounter

Machine	Cpu
Machine M	Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
Machine Z	Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz



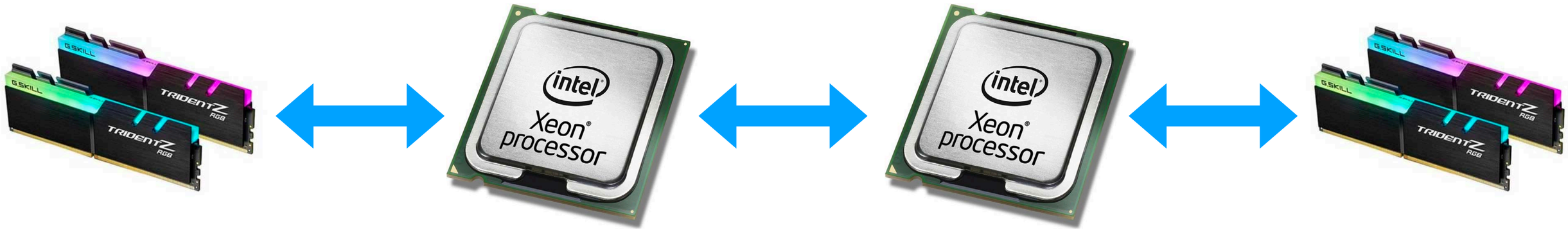
Score for each Cpu broken down by Machine. Color shows details about Benchmark. Shape shows details about Benchmark. Details are shown for Machine. The data is filtered on Threads, which keeps 8. The view is filtered on Benchmark and Machine. The Benchmark filter keeps AtomicDoubleAddBench.viaAtoLo and doubleAccumulator.DoubleAccumulatorBench.withSynchronized. The Machine filter keeps Machine M and Machine Z.

Benchmark

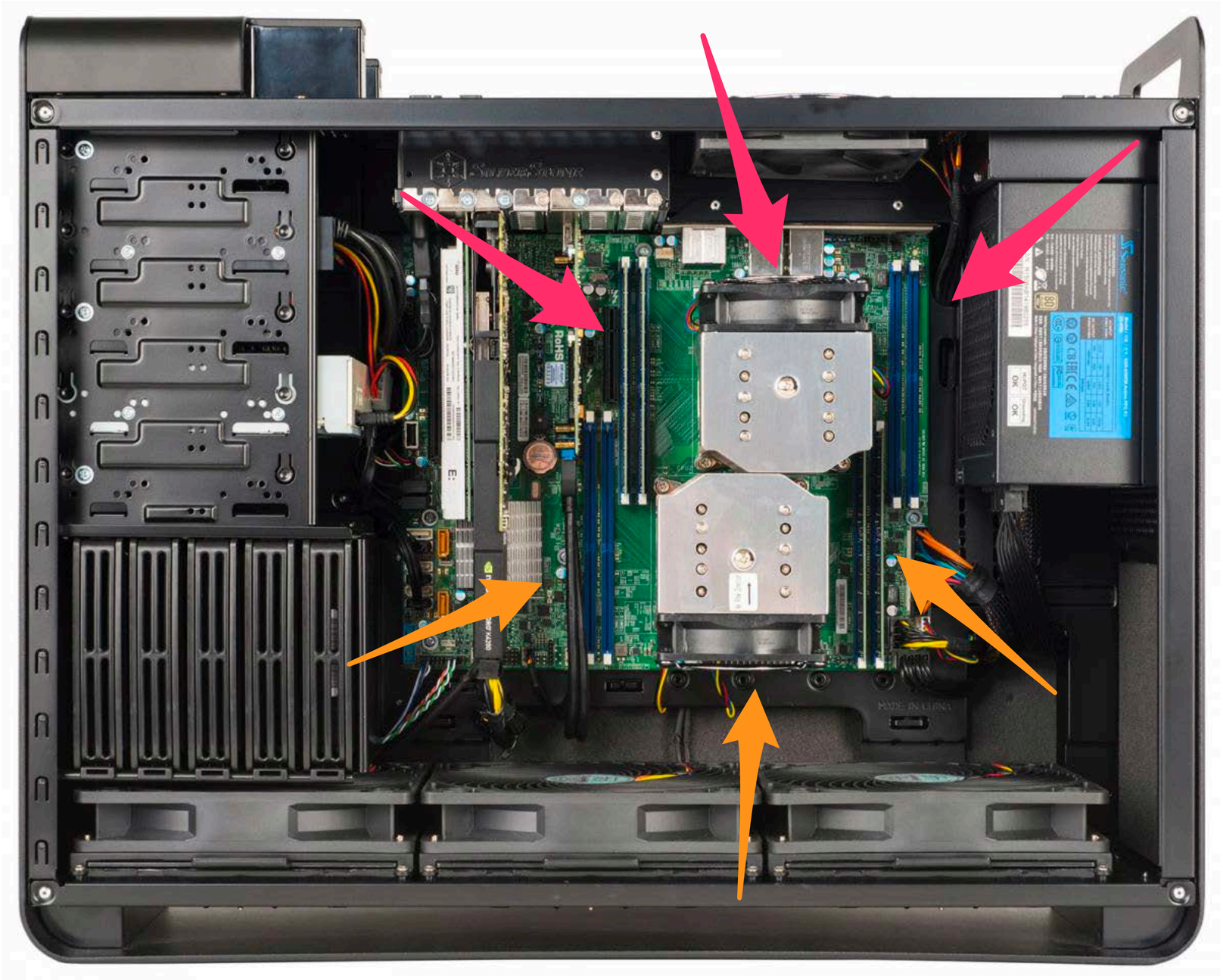
- ○ AtomicDoubleAddBench.viaAtoLo
- ✱ doubleAccumulator.DoubleAccumulatorBench.withSynchronized

Benchmark

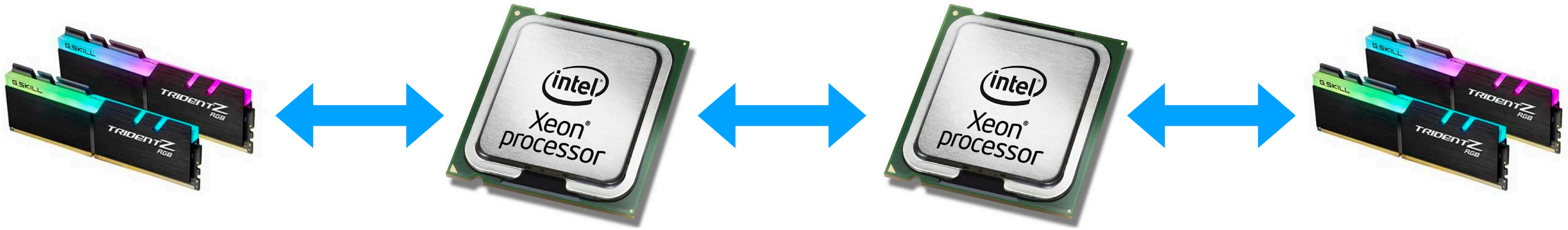
- AtomicDoubleAddBench.viaAtoLo
- doubleAccumulator.DoubleAccumulatorBench.withSynchronized


G.SKILL
TRIDENTZ
RGB
intel
Xeon
processor
cpu0
cpu1


SUPERMICRO
RoHS
NVIDIA QUADRO K420
MADE IN CHINA



```
ags@ags-z6 ~ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 28 29 30 31 32 33 34 35 36 37 38 39 40 41
node 0 size: 64124 MB
node 0 free: 53317 MB
node 1 cpus: 14 15 16 17 18 19 20 21 22 23 24 25 26 27 42 43 44 45 46 47 48 49 50 51 52 53 54 55
node 1 size: 64505 MB
node 1 free: 48525 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```



Option 1: NUMA-friendly code
Option 2: avoid NUMA :-)

AtomicCounter

Machine	Cpu
Machine A	AMD Ryzen Threadripper 2990WX 32-Core Processor
Machine B	AMD EPYC 7551 32-Core Processor

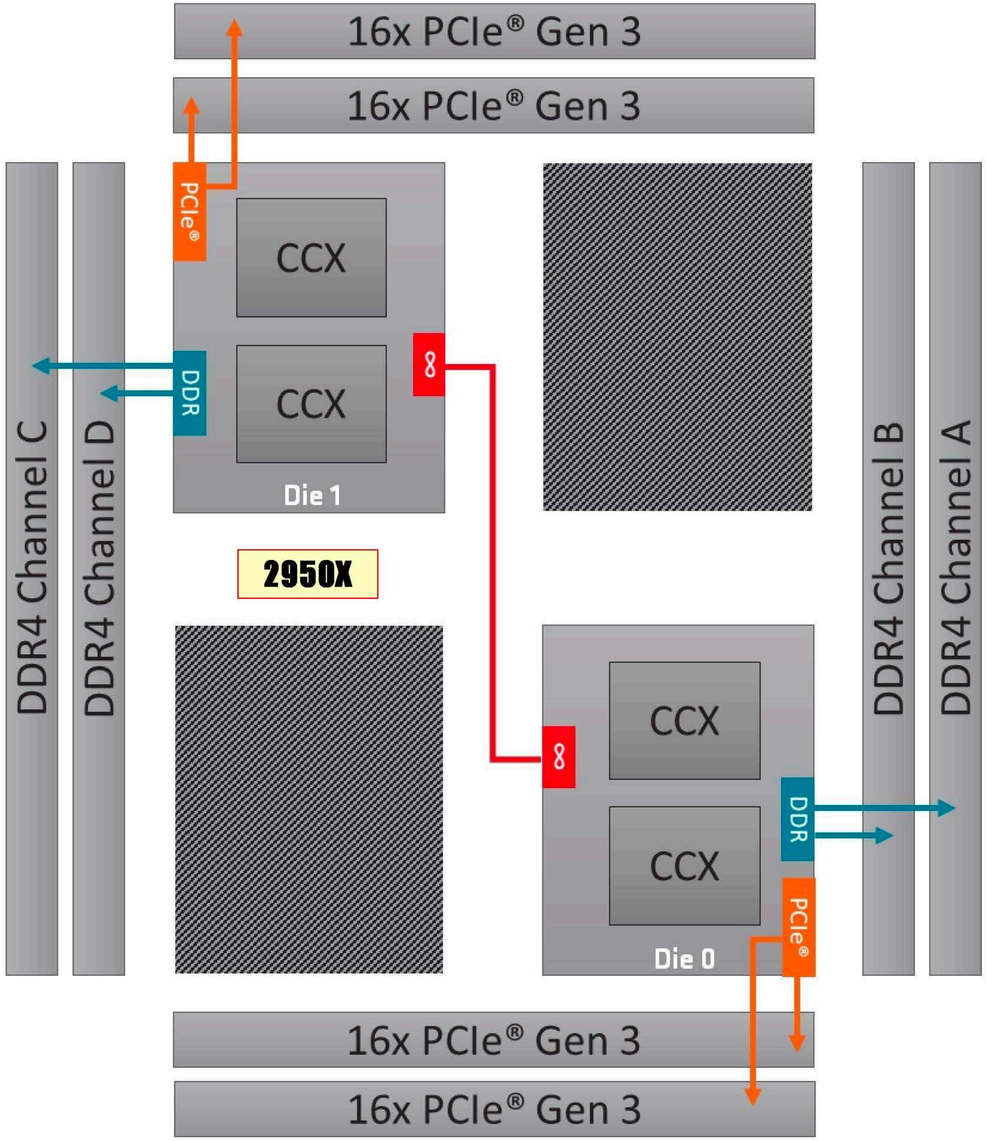
Score for each Cpu broken down by Machine. Color shows details about Benchmark. Shape shows details about Benchmark. Details are shown for Machine. The data is filtered on Threads, which keeps 56, 64 and 80. The view is filtered on Benchmark, Machine and Exclusions (Benchmark,Cpu,Machine,Score). The Benchmark filter keeps AtomicDoubleAddBench.viaAtoLo, AtomicDoubleAddBench.viaGuava, doubleAccumulator.DoubleAccumulatorBench.guava, doubleAccumulator.DoubleAccumulatorBench.jucAtomicFieldUpdater and doubleAccumulator.DoubleAccumulatorBench.jucAtomicLongAsDouble. The Machine filter keeps Machine A and Machine B. The Exclusions (Benchmark,Cpu,Machine,Score) filter keeps 89 members.

Benchmark

- AtomicDoubleAddBench.viaAtoLo
- AtomicDoubleAddBench.viaGuava
- doubleAccumulator.DoubleAccumulatorBench.guava
- doubleAccumulator.DoubleAccumulatorBench.jucAtomicFieldUpdater
- doubleAccumulator.DoubleAccumulatorBench.jucAtomicLongAsDouble

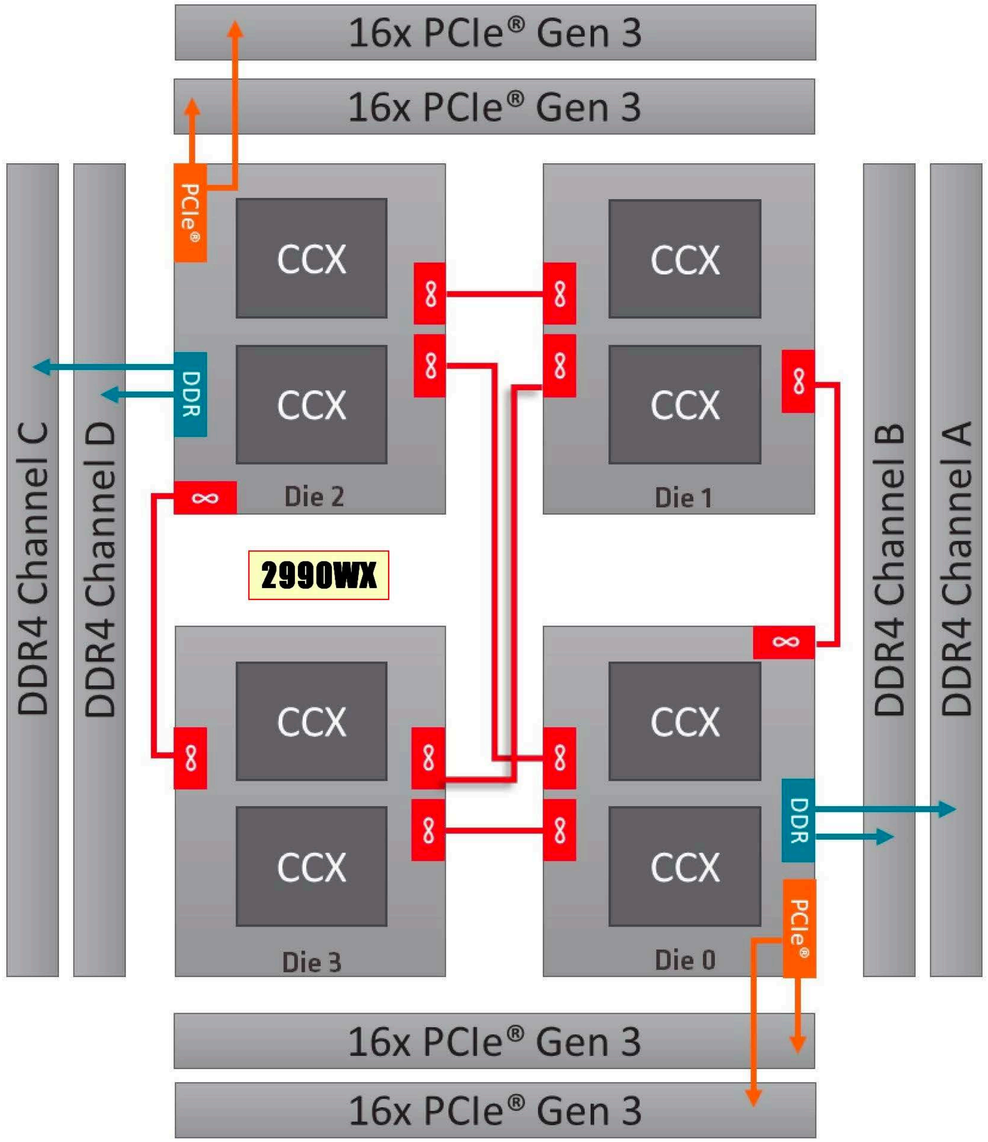
Benchmark

- AtomicDoubleAddBench.viaAtoLo
- AtomicDoubleAddBench.viaGuava
- doubleAccumulator.DoubleAccumulatorBench.guava
- doubleAccumulator.DoubleAccumulatorBench.jucAtomicFieldUpdater
- doubleAccumulator.DoubleAccumulatorBench.jucAtomicLongAsDouble

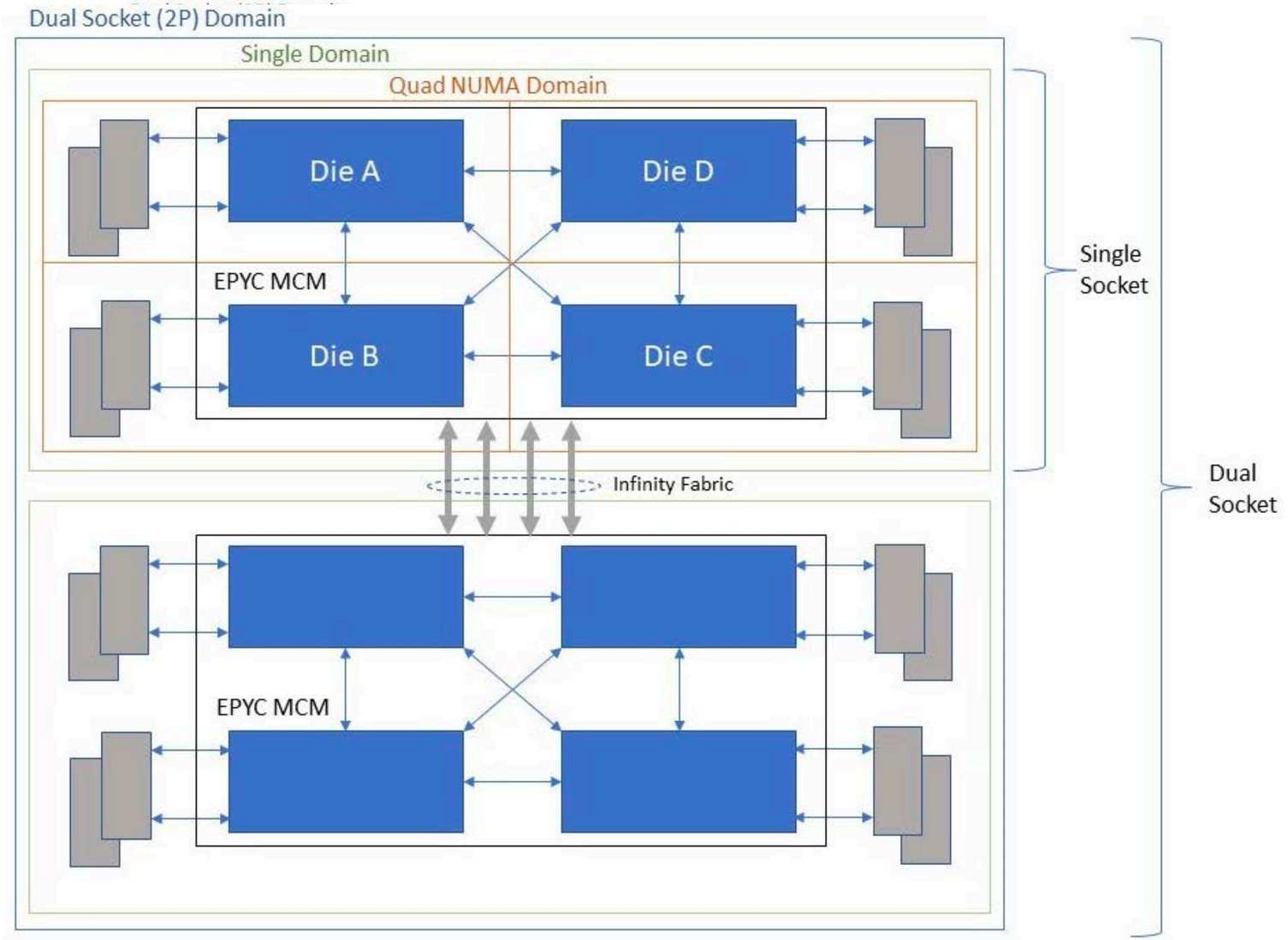

16x PCIe® Gen 3
16x PCIe® Gen 3
PCIe®
CCX
CCX
DDR
Die 1
DDR4 Channel C
DDR4 Channel D
2950X
DDR4 Channel B
DDR4 Channel A
CCX
CCX
DDR
PCIe®
Die 0
16x PCIe® Gen 3
16x PCIe® Gen 3

2950x

2990wx

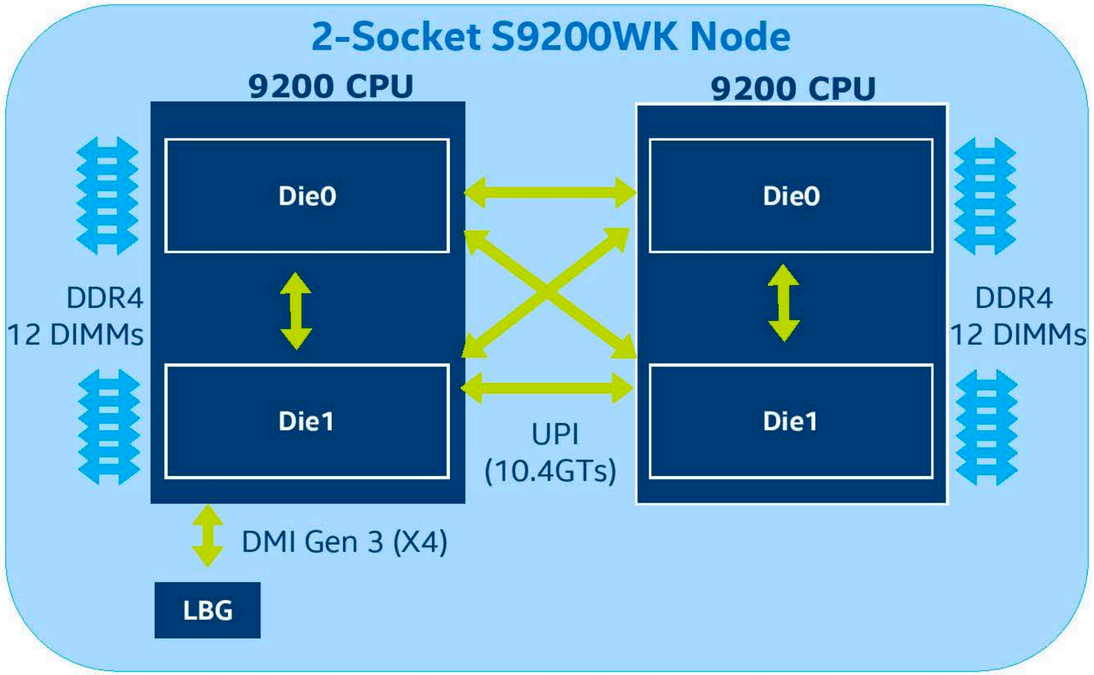

16x PCIe® Gen 3
16x PCIe® Gen 3
PCIe®
DDR
CCX
CCX
CCX
CCX
Die 2
Die 1
2990WX
CCX
CCX
CCX
CCX
Die 3
Die 0
DDR
PCIe®
DDR4 Channel C
DDR4 Channel D
DDR4 Channel B
DDR4 Channel A
16x PCIe® Gen 3
16x PCIe® Gen 3

Dual socket EPYC



https://www.amd.com/system/files/2018-03/AMD-Optimizes-EPYC-Memory-With-NUMA.pdf

Intel® Xeon® Platinum 9200 Processor Overview

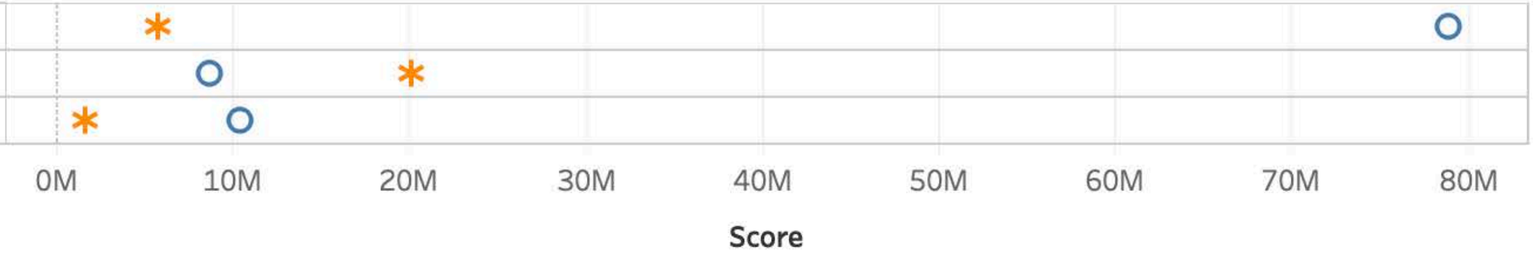


- Intel® Xeon® Platinum 9200 Processors consist of two die in a BGA package
- Multi-chip processor with single hop latency for any of the CPU die to memory in a 2S node
- Key IO/mem features include:
  - 12 ch DDR4 2933 MT/s per CPU
  - 4 UPI x20 wide at 10.4GTs per CPU
  - x80 PCIe G3 lanes per 2S Node in Intel® Server Systems S9200WK*

* Intel® Server Systems S9200WK supports 2 x16 Gen3 slots (per 1U node); 4 x16 Gen3 slots (per 2U node)

AtomicCounter

Machine	Cpu
Machine A	AMD Ryzen Threadripper 2990WX 32-Core Processor
Machine M	Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
Machine Z	Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz



Score for each Cpu broken down by Machine. Color shows details about Benchmark. Shape shows details about Benchmark. Details are shown for Machine. The data is filtered on Threads, which keeps 8. The view is filtered on Benchmark and Machine. The Benchmark filter keeps AtomicDoubleAddBench.viaAtoLo and doubleAccumulator.DoubleAccumulatorBench.withSynchronized. The Machine filter keeps Machine A, Machine B, Machine M and Machine Z.

Benchmark

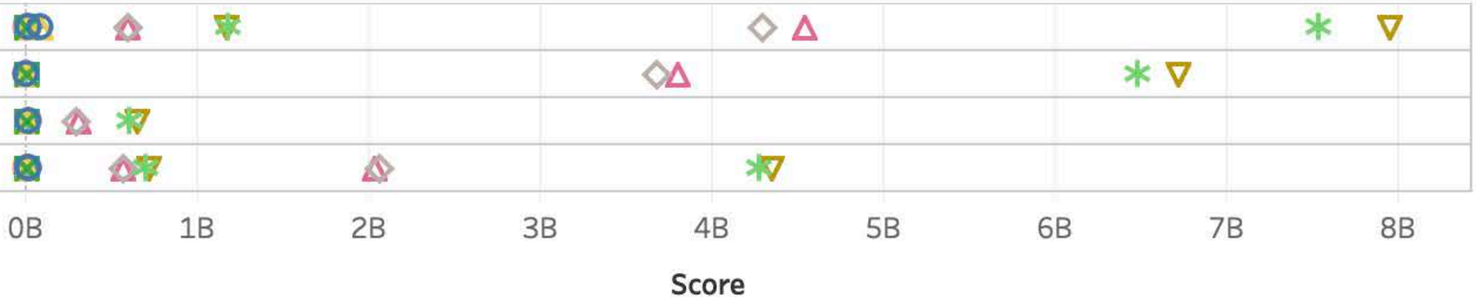
- AtomicDoubleAddBench.viaAtoLo
- doubleAccumulator.DoubleAccumulatorBench.withSynchronized

Benchmark

- AtomicDoubleAddBench.viaAtoLo
- doubleAccumulator.DoubleAccumulatorBench.withSynchronized

AtomicCounter

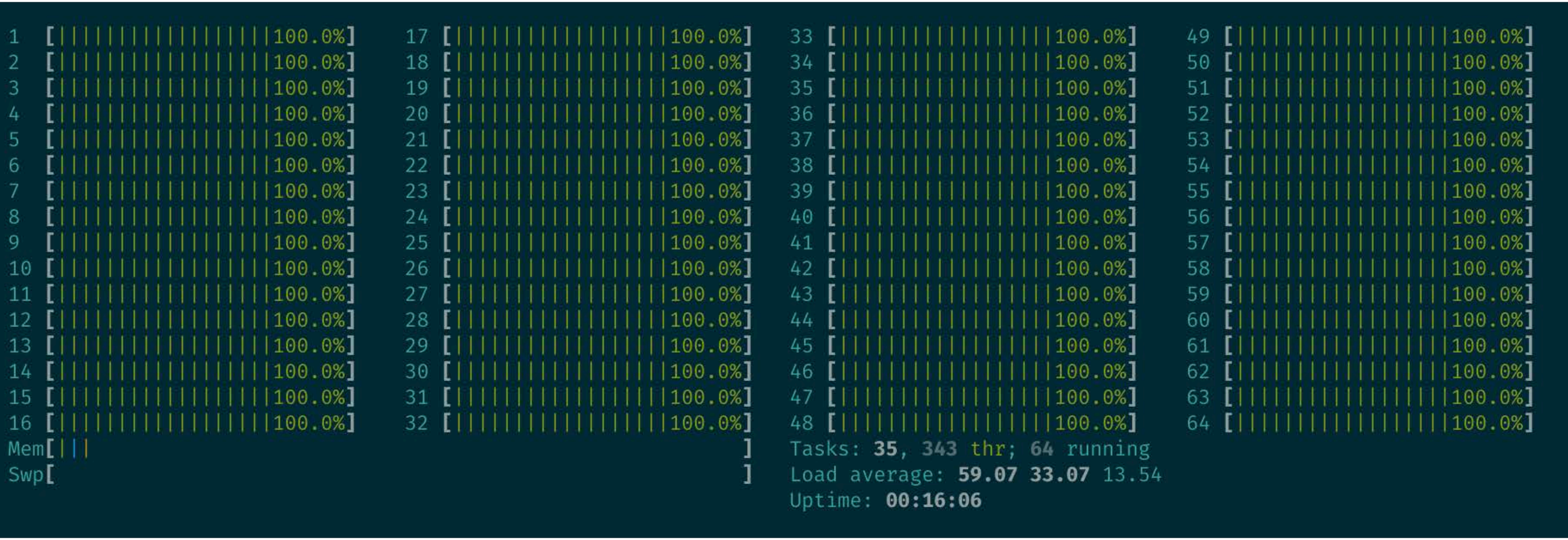
Machine	Cpu
Machine A	AMD Ryzen Threadripper 2990WX 32-Core Processor
Machine B	AMD EPYC 7551 32-Core Processor
Machine M	Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
Machine Z	Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz



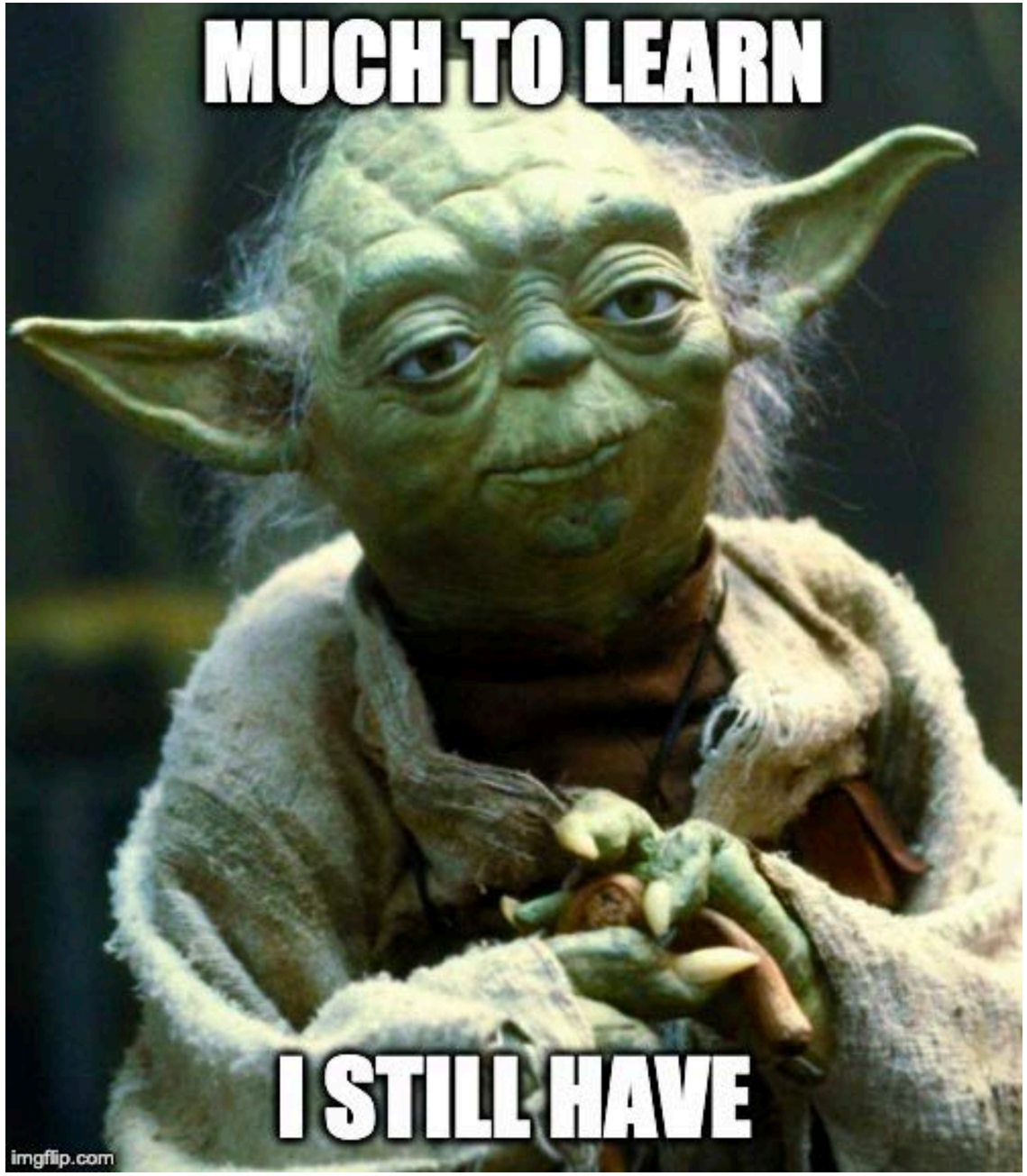
Score for each Cpu broken down by Machine. Color shows details about Benchmark. Shape shows details about Benchmark. Details are shown for Machine. The data is filtered on Threads, which keeps 8, 56, 64 and 80. The view is filtered on Benchmark and Machine. The Benchmark filter excludes AtomicDoubleAddBench.viaAtomicFieldUpdater_CAS and AtomicDoubleAddBench.viaAtomicFieldUpdater_weakCAS. The Machine filter keeps Machine A, Machine B, Machine M and Machine Z.

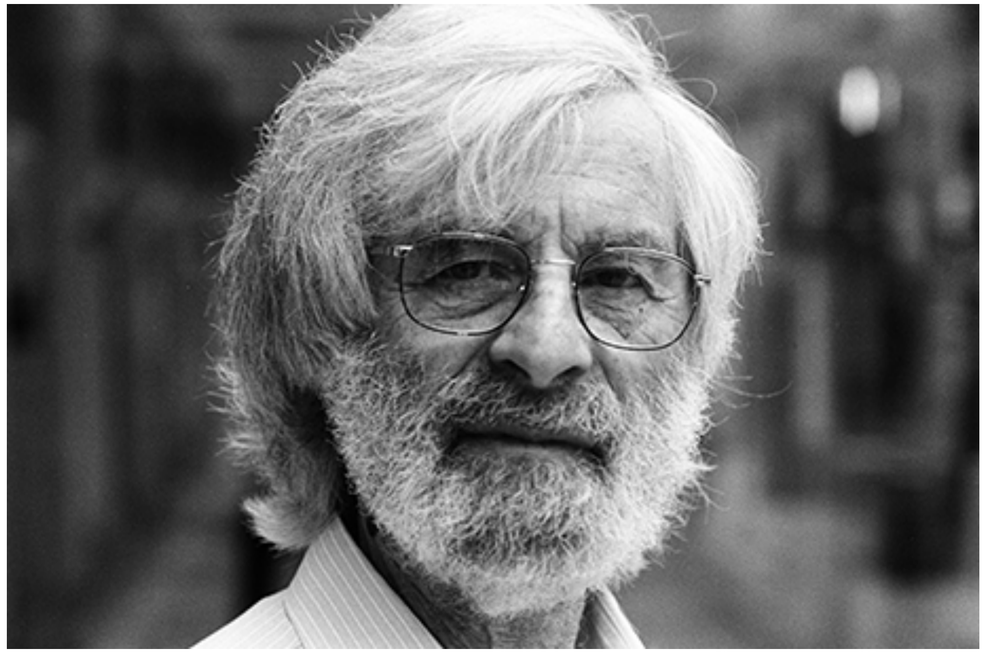
Benchmark

- ○ AtomicDoubleAddBench.viaAtoLo
- ✕ AtomicDoubleAddBench.viaAtoRef
- ✱ AtomicDoubleAddBench.viaDoubleAcc
- ▽ AtomicDoubleAddBench.viaDoubleAdder
- △ AtomicDoubleAddBench.viaGuava
- □ AtomicDoubleAddBench.viaWeakAtoRef
- \+ doubleAccumulator.DoubleAccumulatorBench.guava
- ✕ doubleAccumulator.DoubleAccumulatorBench.jucAtomicFieldUpdater
- ○ doubleAccumulator.DoubleAccumulatorBench.jucAtomicLongAsDouble
- □ doubleAccumulator.DoubleAccumulatorBench.jucAtomicReference
- ◇ doubleAccumulator.DoubleAccumulatorBench.jucDoubleAccumulator
- △ doubleAccumulator.DoubleAccumulatorBench.jucDoubleAdder
- ✱ doubleAccumulator.DoubleAccumulatorBench.withSynchronized


1 [||||||||||||||||||100.0%]
2 [||||||||||||||||||100.0%]
3 [||||||||||||||||||100.0%]
4 [||||||||||||||||||100.0%]
5 [||||||||||||||||||100.0%]
6 [||||||||||||||||||100.0%]
7 [||||||||||||||||||100.0%]
8 [||||||||||||||||||100.0%]
9 [||||||||||||||||||100.0%]
10 [||||||||||||||||||100.0%]
11 [||||||||||||||||||100.0%]
12 [||||||||||||||||||100.0%]
13 [||||||||||||||||||100.0%]
14 [||||||||||||||||||100.0%]
15 [||||||||||||||||||100.0%]
16 [||||||||||||||||||100.0%]
17 [||||||||||||||||||100.0%]
18 [||||||||||||||||||100.0%]
19 [||||||||||||||||||100.0%]
20 [||||||||||||||||||100.0%]
21 [||||||||||||||||||100.0%]
22 [||||||||||||||||||100.0%]
23 [||||||||||||||||||100.0%]
24 [||||||||||||||||||100.0%]
25 [||||||||||||||||||100.0%]
26 [||||||||||||||||||100.0%]
27 [||||||||||||||||||100.0%]
28 [||||||||||||||||||100.0%]
29 [||||||||||||||||||100.0%]
30 [||||||||||||||||||100.0%]
31 [||||||||||||||||||100.0%]
32 [||||||||||||||||||100.0%]
33 [||||||||||||||||||100.0%]
34 [||||||||||||||||||100.0%]
35 [||||||||||||||||||100.0%]
36 [||||||||||||||||||100.0%]
37 [||||||||||||||||||100.0%]
38 [||||||||||||||||||100.0%]
39 [||||||||||||||||||100.0%]
40 [||||||||||||||||||100.0%]
41 [||||||||||||||||||100.0%]
42 [||||||||||||||||||100.0%]
43 [||||||||||||||||||100.0%]
44 [||||||||||||||||||100.0%]
45 [||||||||||||||||||100.0%]
46 [||||||||||||||||||100.0%]
47 [||||||||||||||||||100.0%]
48 [||||||||||||||||||100.0%]
49 [||||||||||||||||||100.0%]
50 [||||||||||||||||||100.0%]
51 [||||||||||||||||||100.0%]
52 [||||||||||||||||||100.0%]
53 [||||||||||||||||||100.0%]
54 [||||||||||||||||||100.0%]
55 [||||||||||||||||||100.0%]
56 [||||||||||||||||||100.0%]
57 [||||||||||||||||||100.0%]
58 [||||||||||||||||||100.0%]
59 [||||||||||||||||||100.0%]
60 [||||||||||||||||||100.0%]
61 [||||||||||||||||||100.0%]
62 [||||||||||||||||||100.0%]
63 [||||||||||||||||||100.0%]
64 [||||||||||||||||||100.0%]
Mem[|||
Swp[
Tasks: 35, 343 thr; 64 running
Load average: 59.07 33.07 13.54
Uptime: 00:16:06

The times are interesting


MUCH TO LEARN
I STILL HAVE
imgflip.com



A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Leslie Lamport

https://www.microsoft.com/en-us/research/publication/distribution/

Taking a step back

Cluster Network IO

Query: select total_bytes_receive_rate_across_network_interfaces, total_bytes_transmit_rate_across_network_interfaces where category = CLUSTER

Data Granularity: Auto

Legend | Details | Distribution

- ✔ Total Bytes Received Across Network Interfaces
- ✔ Total Bytes Transmitted Across Network Interfaces

Cluster Network IO

Query: select total_bytes_receive_rate_across_network_interfaces, total_bytes_transmit_rate_across_network_interfaces where category = CLUSTER

Data Granularity: Auto

~100 MB/s

Legend | Details | Distribution

- ✔ Total Bytes Received Across Network Interfaces
- ✔ Total Bytes Transmitted Across Network Interfaces


Top of Rack (ToR)
(Rear of servers shown for clarity)

TCP incast

http://bradhedlund.com/2011/05/01/tcp-incast-and-cloud-application-performance/


UBNT-UF-MM-1G
UBNT-UF-MM-1G

Serialisation

Let’s consider a Game of Life with population of the UK

What works for me:

numactl bound processes, talking over localhost

What works?

Serial

(or sequential, or vectorised)

Event Sourcing

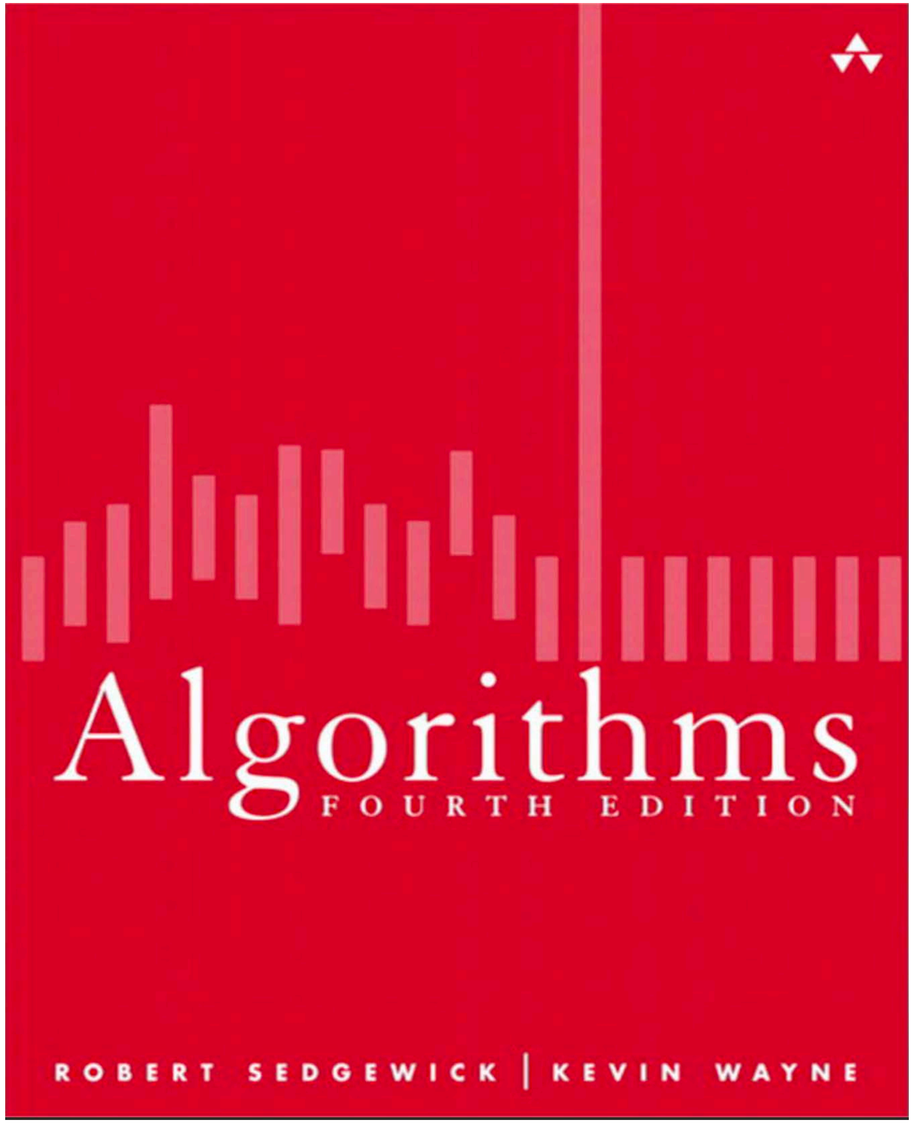
Events

Conclusions:

Get basics right

Keep it simple

The Morning Paper


Algorithms
FOURTH EDITION
ROBERT SEDGEWICK | KEVIN WAYNE

Cut, divide, simplify

The best way of locking

is not to lock

buy you a 2cpu, even if ebay

Define *your* criteria

Analyse the problem as a whole

Make hypothesis,
verify,
distrust results

```
[info] REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on
[info] why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial
[info] experiments, perform baseline and negative tests that provide experimental control, make sure
[info] the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts.
[info] Do not assume the numbers tell you what you want them to tell.
```

Ceterum autem censeo
Carthaginem esse delendam