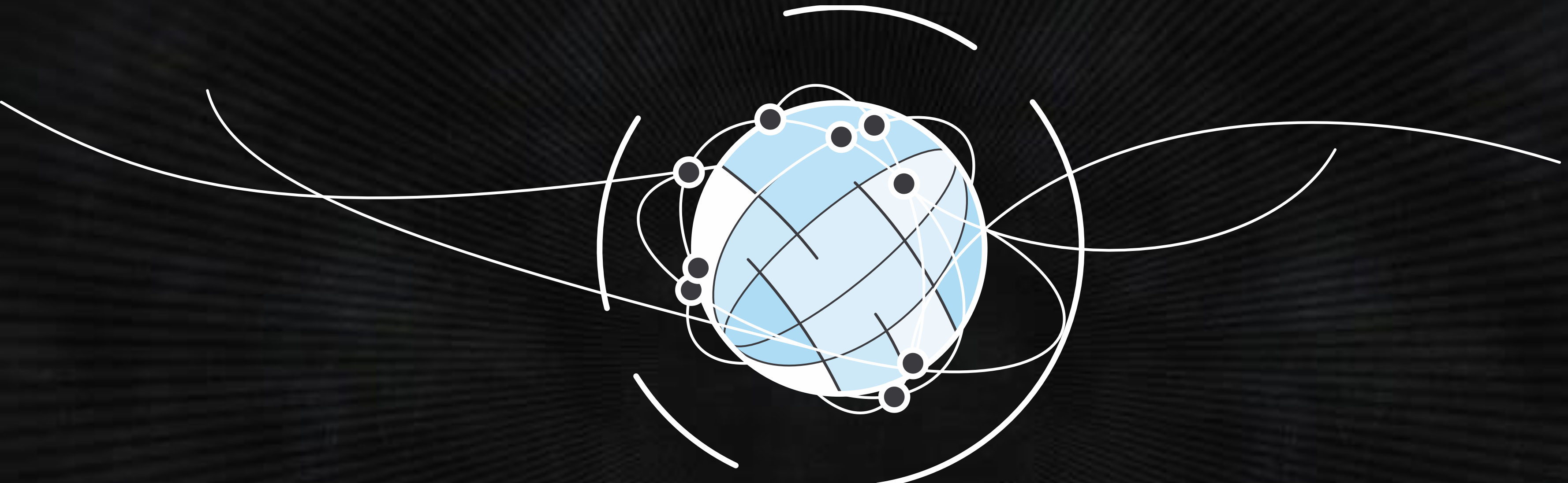
Scaling Tightly Coupled Algorithms on AWS



Dr. Scott Eberhardt

Principle Solutions Architect – HPC, AWS

Visiting Reader, Imperial College

aws

Research Computing @ AWS

AWS Worldwide Research & Technical Computing

IT'S ABOUT SCIENCE, NOT SERVERS.

aws.amazon.com/rcp

#AWSresearchcloud

Unlimited infrastructure

Low cost with flexible pricing

Efficient clusters

Why AWS for HPC?

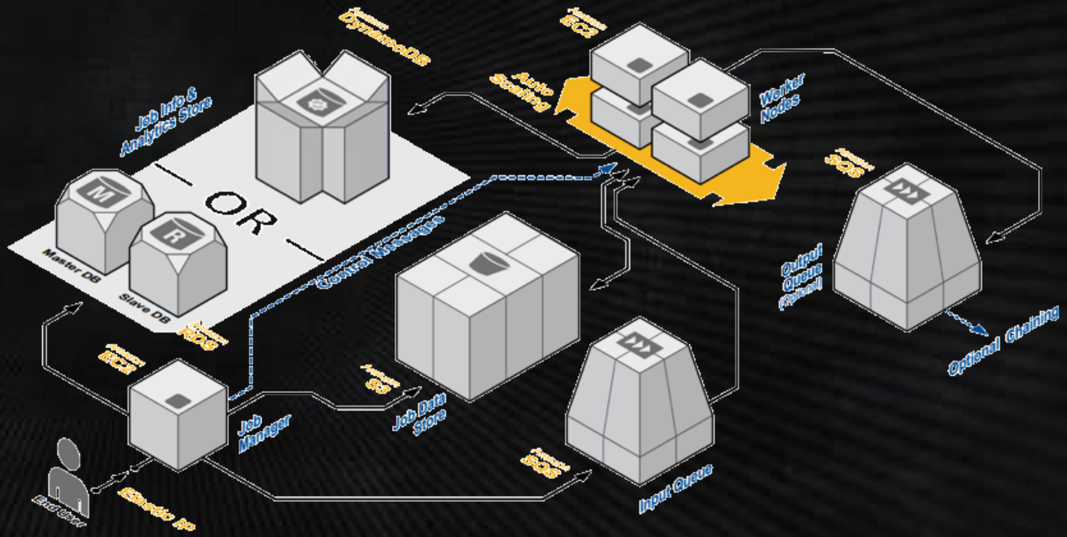
Faster time to results

Increased collaboration

Concurrent clusters on-demand

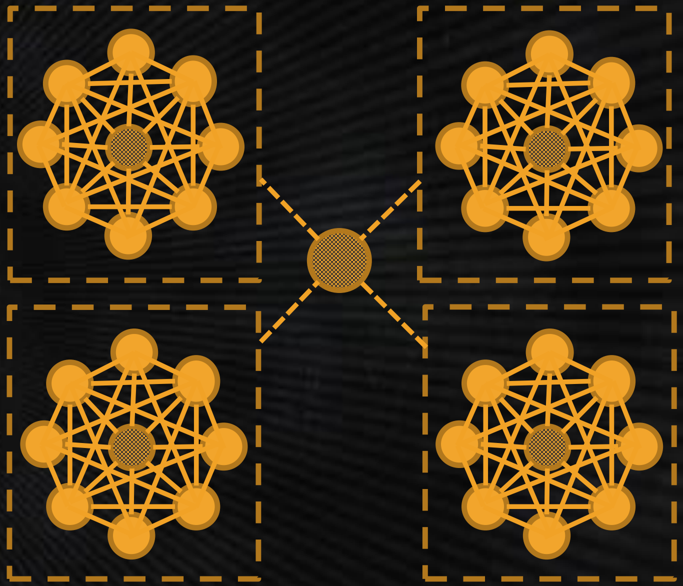
Great Features for HPC Workloads

- Experimentation without Fear!
- Start and stop instances!
- Spot Pricing
- Continuous Updates
  - Compute
  - Network
  - Storage
  - Services



Cloud Improves Workload Throughput

- Run many Jobs in Parallel
  - Eliminate HPC resource contention
  - Eliminate queue wait
  - Use it when you need it
- Right-size clusters and resources
  - Optimize each workload for performance
  - Pay for only what you use

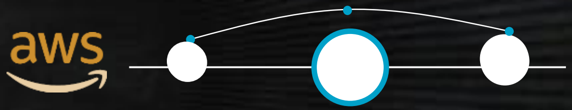


Cost advantages

On Premises
Capital Expense Model

- High upfront capital cost
- High cost of ongoing support

Amazon Web Services
Pay As You Go Model



- Use only what you need
- Multiple pricing models

Popular HPC workloads on AWS

Genomics Processing

Modeling and Simulation

Government and Educational Research

Monte Carlo Simulations

Transcoding and Encoding

Computational Chemistry

… and many more

Defining HPC – example use cases

Clustered (Tightly coupled)

Data Light

Minimal requirements for high performance storage

Data Heavy

Benefits from access to high performance storage

Clustered / Data Light:
- Fluid dynamics
- Weather forecasting
- Materials simulations
- Crash simulations

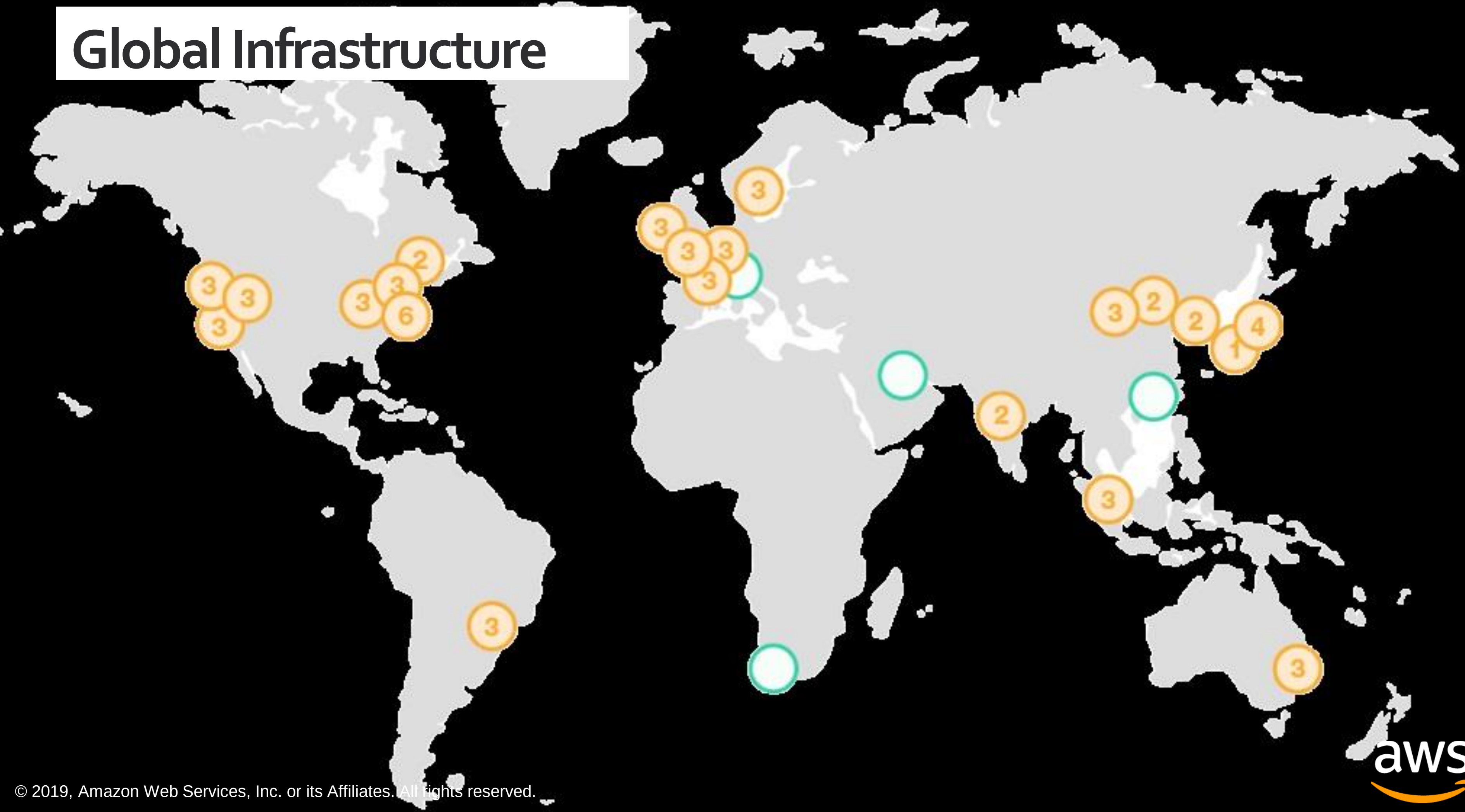
Clustered / Data Heavy:
- Seismic processing
- Metagenomics
- Astrophysics
- Deep learning

Distributed / Data Light:
- Risk modeling
- Molecular modeling
- Contextual search
- Logistics simulations

Distributed / Data Heavy:
- Animation and VFX
- Semiconductor verification
- Image processing/GIS
- Genomics

Distributed / Grid (Loosely coupled)

Global Infrastructure



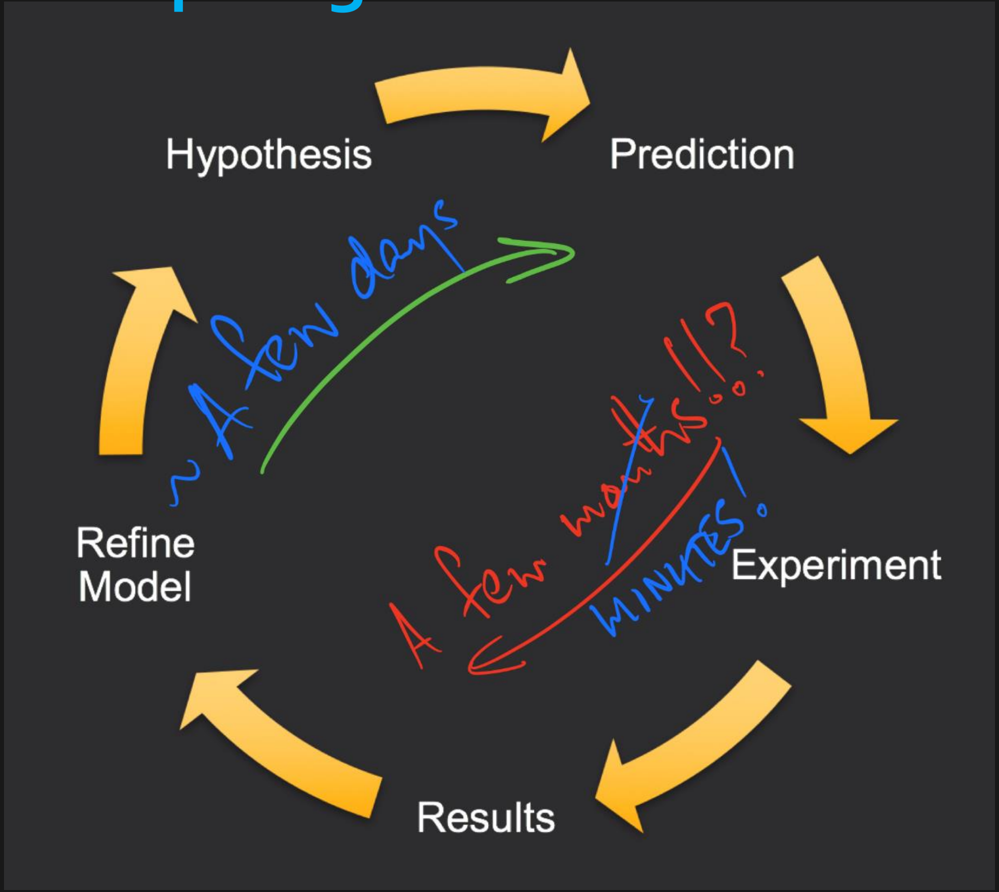
Important enablers for HPC on the cloud

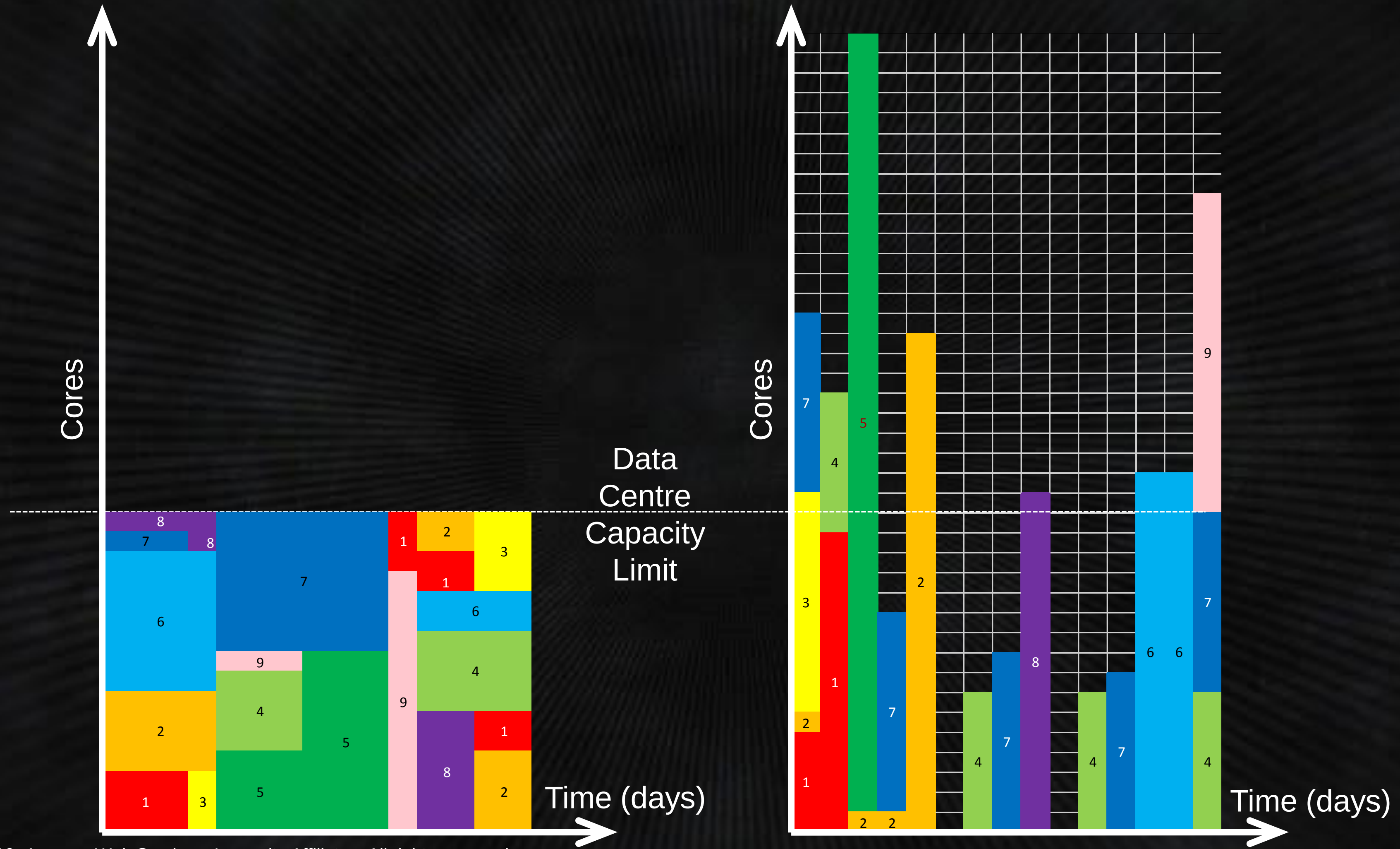
- Compute performance – CPUs, GPUs, FPGAs
- Memory performance – high RAM requirements in many applications
- Network performance – throughput, latency, and consistency
- Storage performance – including shared filesystems
- Automation and cluster/job management
- Remote graphics for interactive applications
- ISV support – including license management

...and SCALE

The Scientific Computing Method

Credit: Aristotle



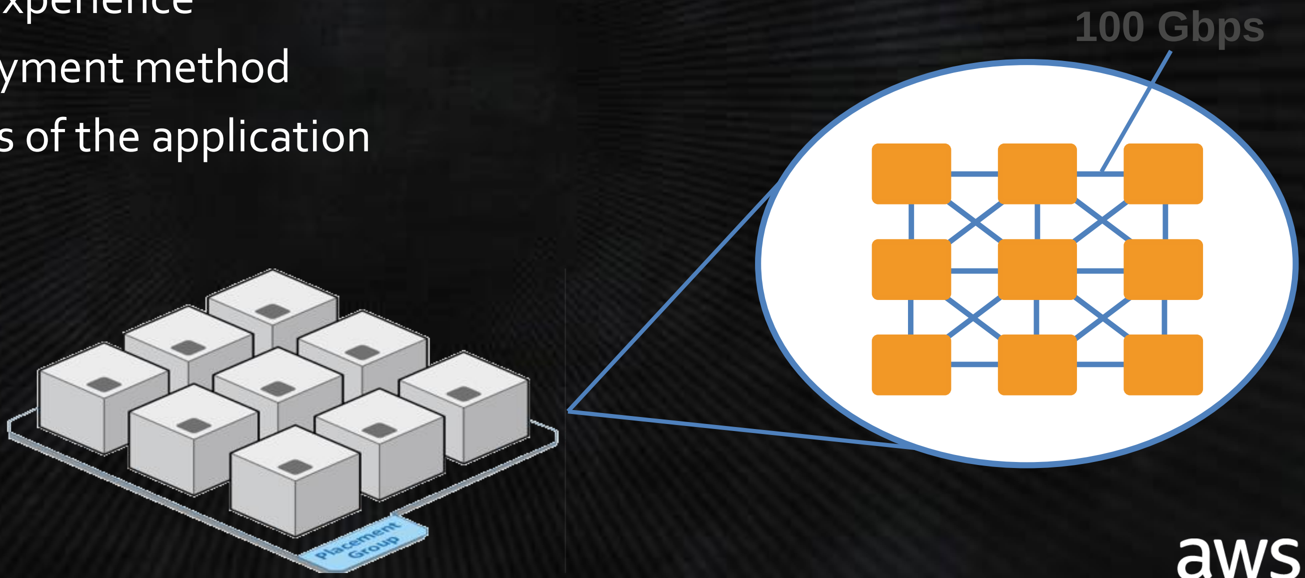

Cores
Time (days)
Data
Centre
Capacity
Limit
Cores
Time (days)

Deploy multiple HPC clusters

Running at the same time, and tuned for each workload

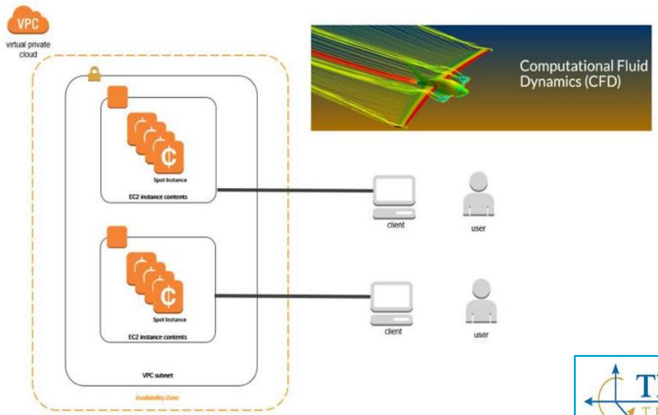
Which architecture Do I choose?

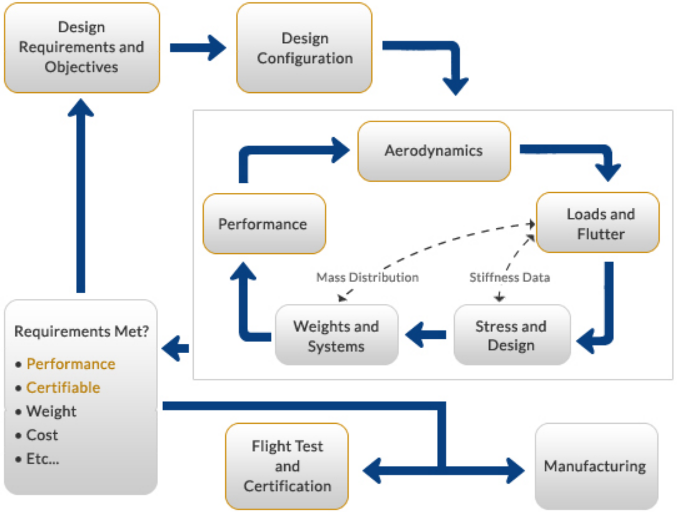
- One size does not fit all
- Architectures are an opportunity for optimization
- The chosen architecture depends on:
  - The desired user experience
  - The desired deployment method
  - The characteristics of the application



Example in aerospace

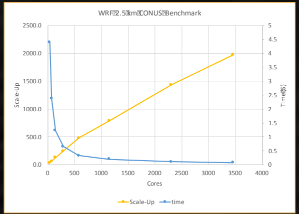
- Running parallel CFD studies using Siemens STAR-CCM+
  - Goal: shorten the time between Design Requirements and Configuration, and Flight Testing
- 1000+ cores per CFD study, multiple studies required for each workflow iteration
- Job-level optimizations:
  - Enhanced Networking, Placement Groups
  - Amazon Linux, Hyper-threading disabled
- Workflow optimizations:
  - Spot instances, multiple clusters
  - Multiple parallel studies for faster throughput





Performance considerations

For tightly-coupled cluster workloads



Test using real-world examples

- Use large cases for testing: do not benchmark scalability using only small examples

Domain decomposition

- Choose number of cells per core for either pre-core efficiency or for faster results

Processor states

- Use P-states to reduce processor variability

MPI libraries

- Test with Intel MPI and OpenMPI 4.0, and make use of available tunings

Network

- Use a placement group
- Enable enhanced networking

Hyper-threading and affinity

- Test with Hyper-threading (HT) on and off – usually off is best, but not always
- Use CPU affinity to pin threads to CPU cores when HT is off

Amdahl’s Scaling

What is Amdahl's Law?

$$Speedup = \frac{1}{(1-p)+\frac{p}{n}}$$

p – fraction of code that can be run in parallel
n – system resource improvement (number of processors)

Ideal: $\lim_{n \to \infty} speedup = \frac{1}{1-p}$

Or: $\lim_{p \to 1} speedup = n$

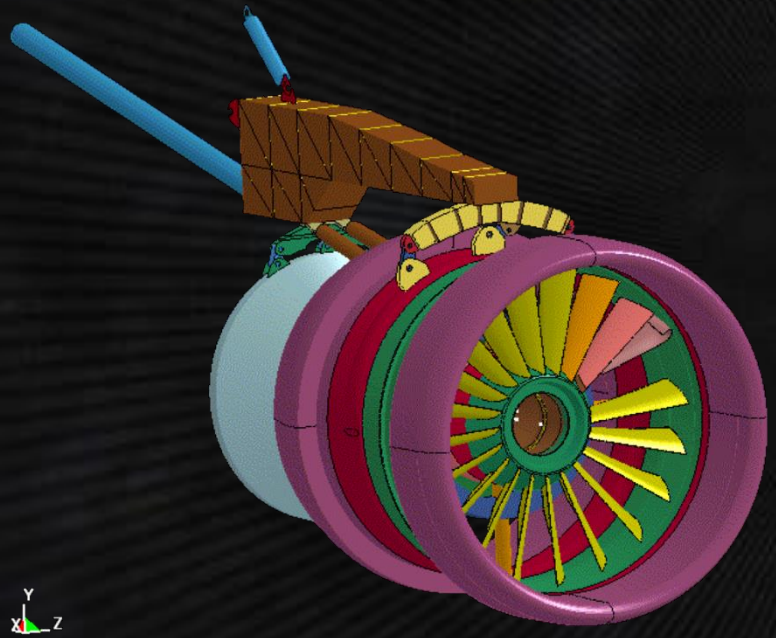
How is it used?

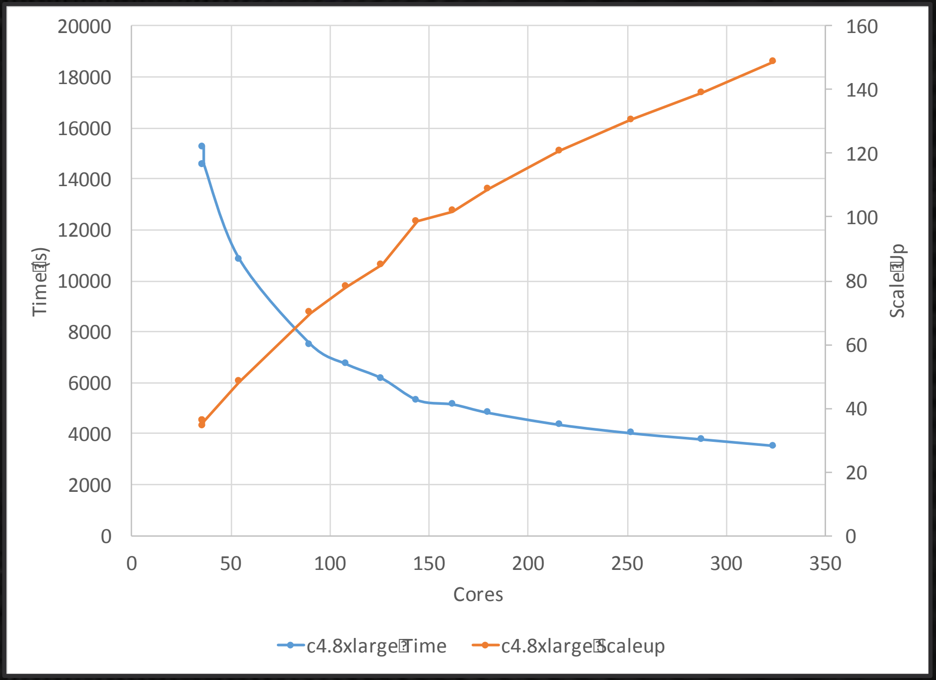
- Amdahl's Law assumes a fixed problem size.
- Example: a CFD calculation with a fixed number of cells – eg. 40M
- As the number of processors increases, the number of CFD mesh cells per compute core decreases.   At some point communication between nodes and cores becomes a bottleneck. This is the driver for low-latency networks.
- Note: The lower the latency the fewer CFD mesh cells per compute node are necessary for good scaling.   Lower latency usually means better speedup when considering Amdahl's law.

Coding Goals

- The following have been routine in code architectures since the first vector computers:
- Choose algorithms where *p* is maximized
-
- Reprogram codes
- Look for ways to avoid/reduce dependencies
- Consider tradeoffs between recalculating and storing
- Monitor code execution to find bottlenecks

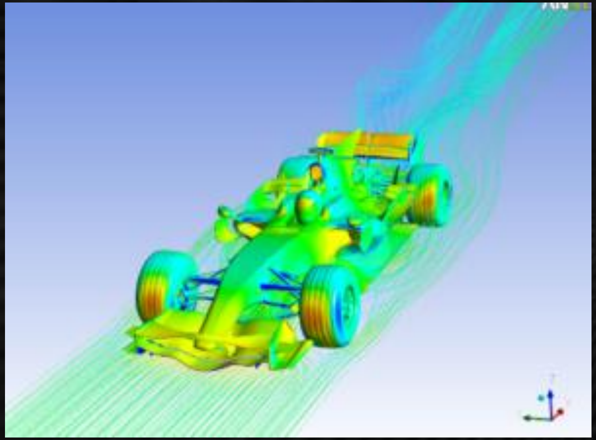
Structural simulation

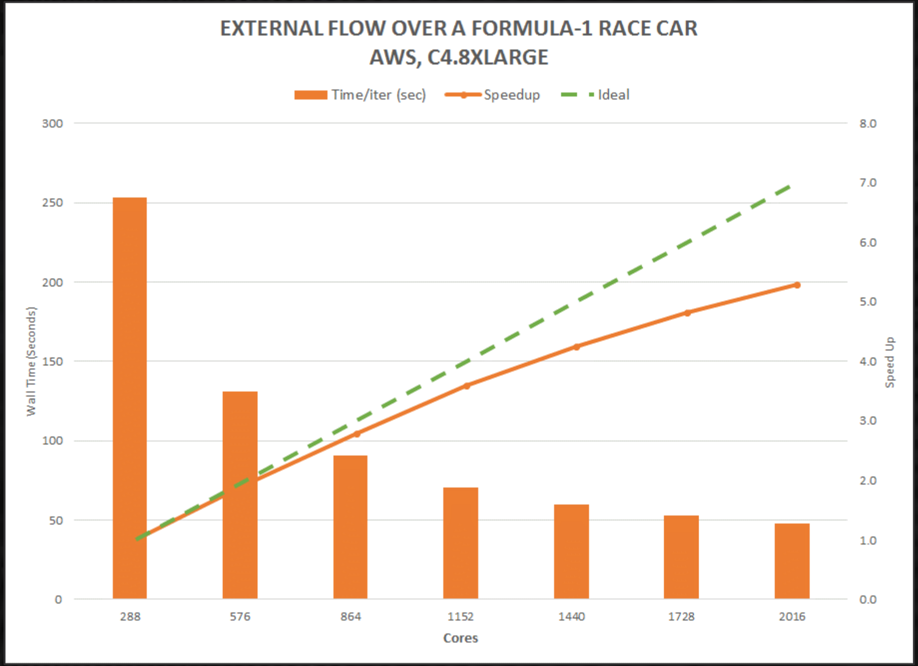




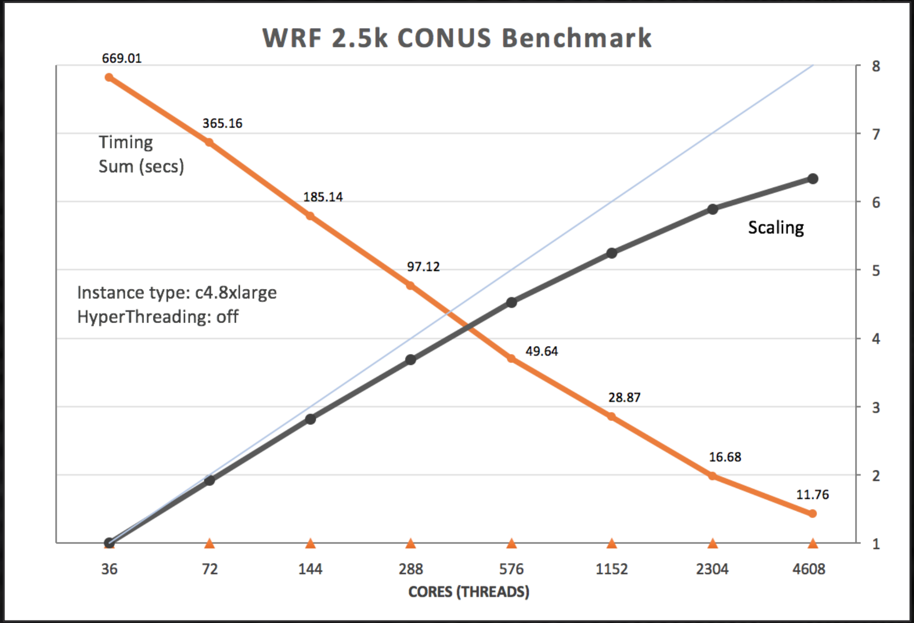
Fluid dynamics – Ansys Fluent

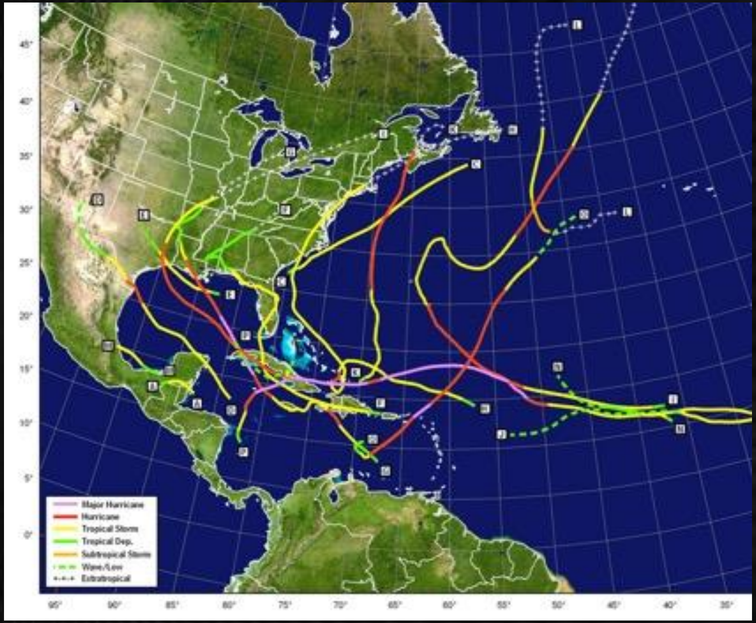
- C4.8xlarge instance type
- 140M cell model
- F1 car CFD benchmark





Tightly-coupled HPC – weather





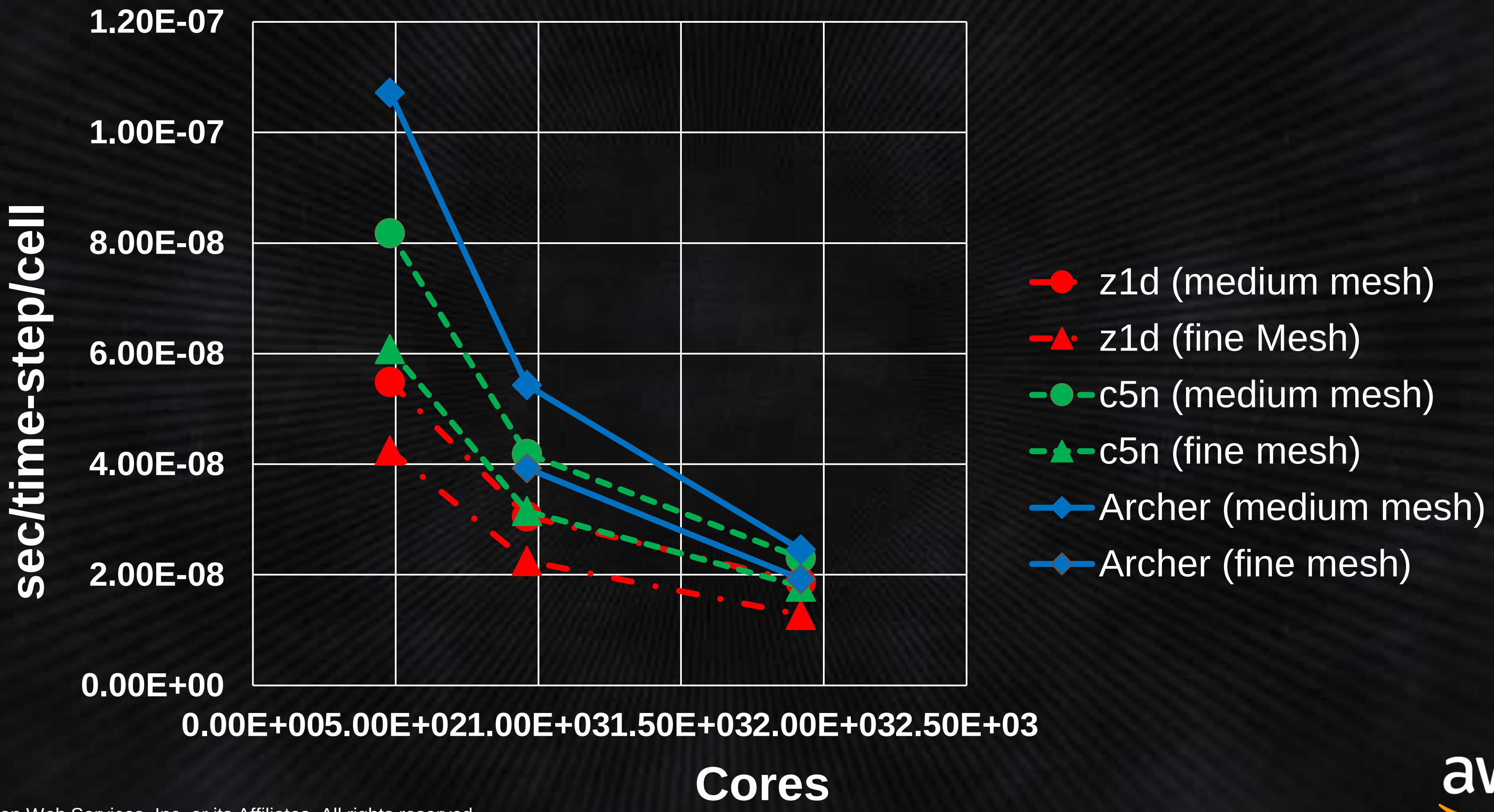
Resources used in this study

- Archer: Cray XC30 supercomputer
  - two 2.7 GHz, 12-core Intel E5-2697 v2 (Ivy Bridge)
- AWS:
  - z1d: 4.0 GHZ Intel®Xeon®Scalable Processors; 24 core per instance; 16GB Ram per core; 25 Gigabit network bandwidth
  - c5n: 3.0/3.5 GHZ Intel®Xeon®Scalable Processors; 36 core per instance; 5.3 GB Ram per core; 100 Gigabit network bandwidth; New Elastic Fabric Adaptor (EFA) for fast networking

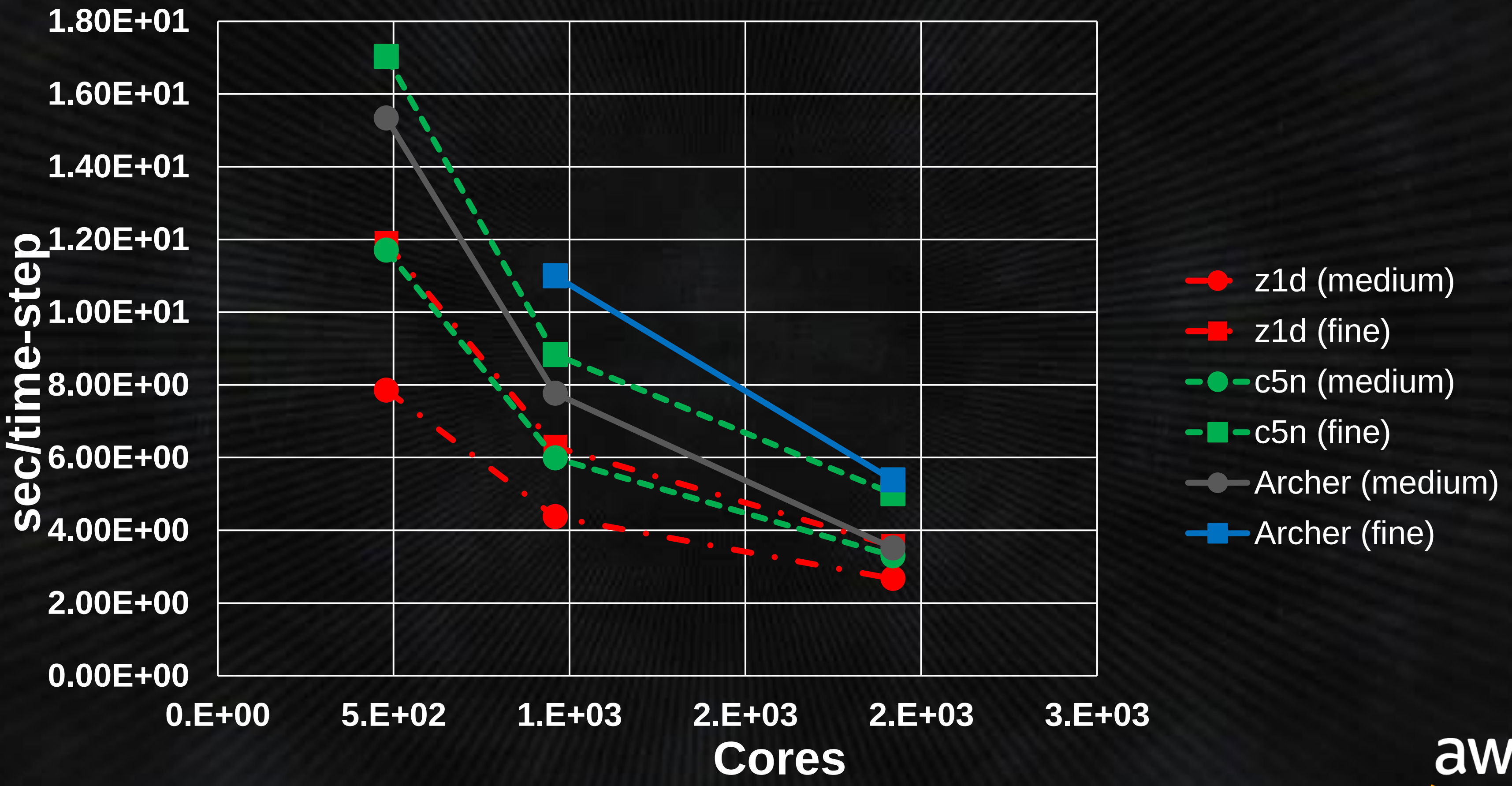
Methodology

- OpenFOAM v1806 in Double Precision (pimpleFoam)
- Scotch decomposition for solving, hierarchical (i.e constant x/y/z loading) for meshing
- SST-DDES Turbulence Model
- ANSA generated 143/280M cell unstructured mesh
- Time Step=5e-^{4}s with 5 inner iterations
- Preconditioned Conjugant Gradient Linear Solver

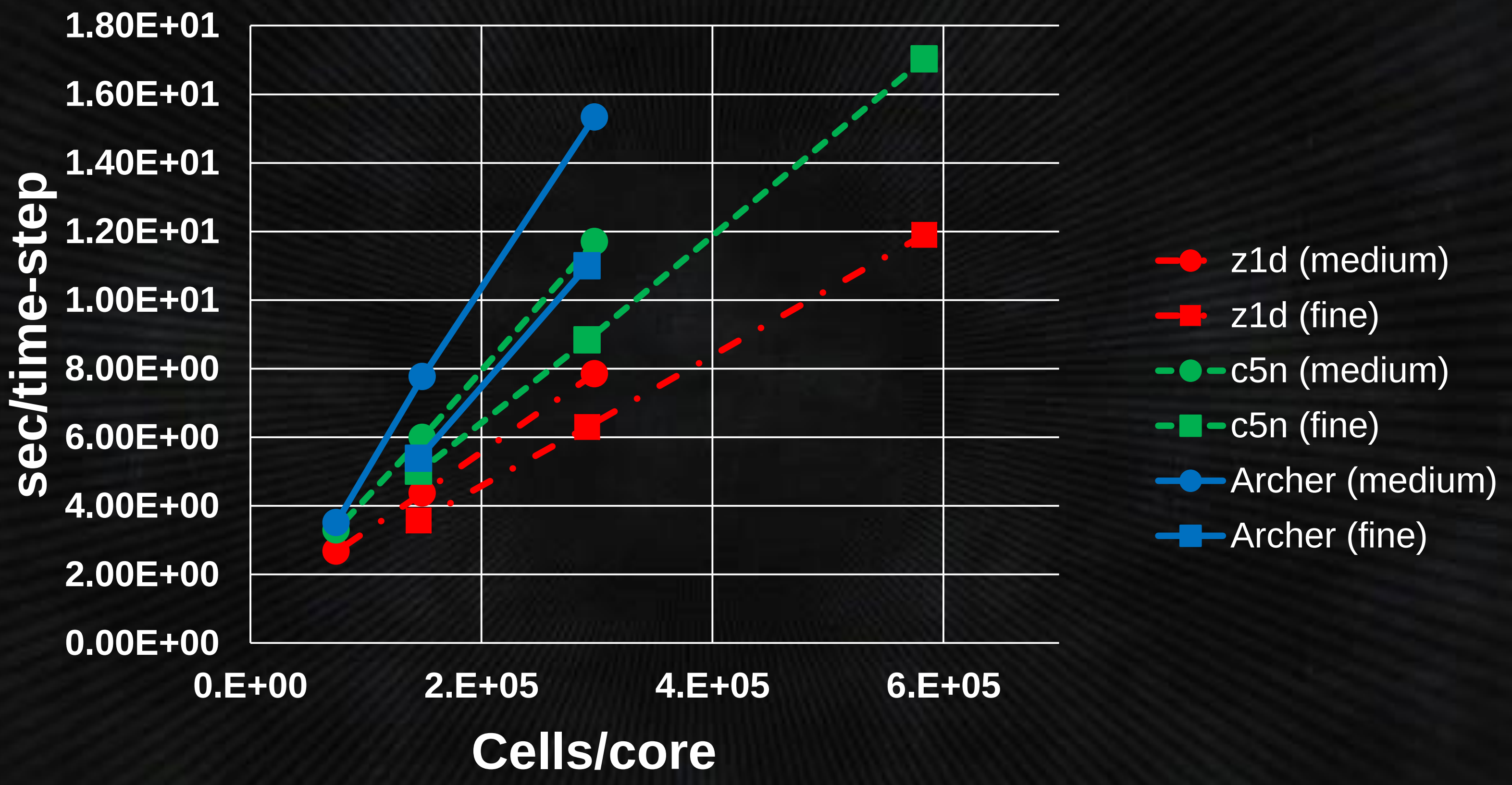
Amdahl Scaling: openFoam (pimpleFoam)



Amdahl Scaling, Alternate View



Amdahl Scaling, 2nd Alternate View



Scaling

Gustafson-Barsis Scaling (aka Gustafson's Law)

What is Gustafson's Law?

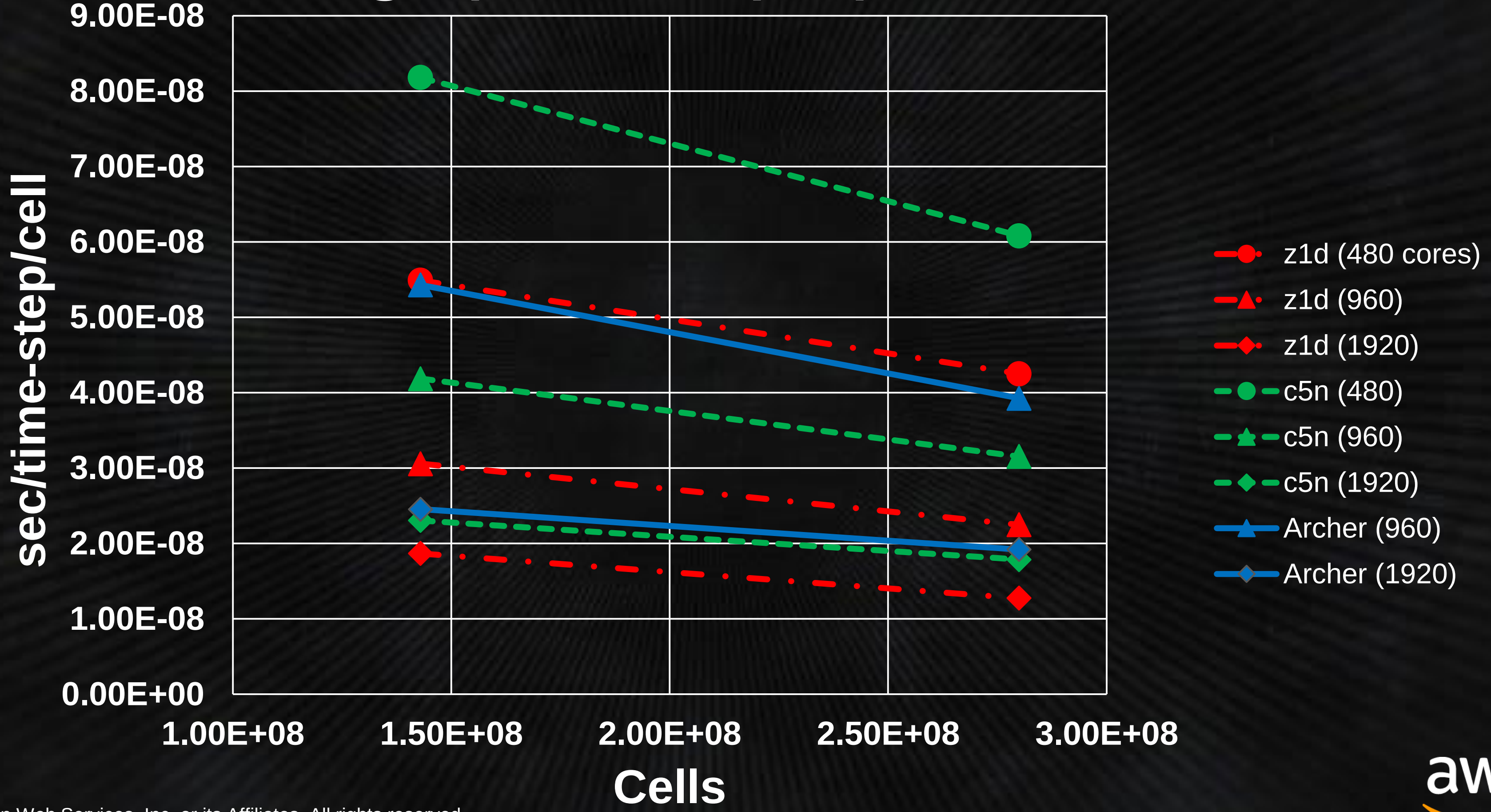
$$Speedup = (1 - p) + \frac{p}{n}$$

Gustafson's Law assumes a workload will increase in size linearly with the number of processors. (scalable workload)

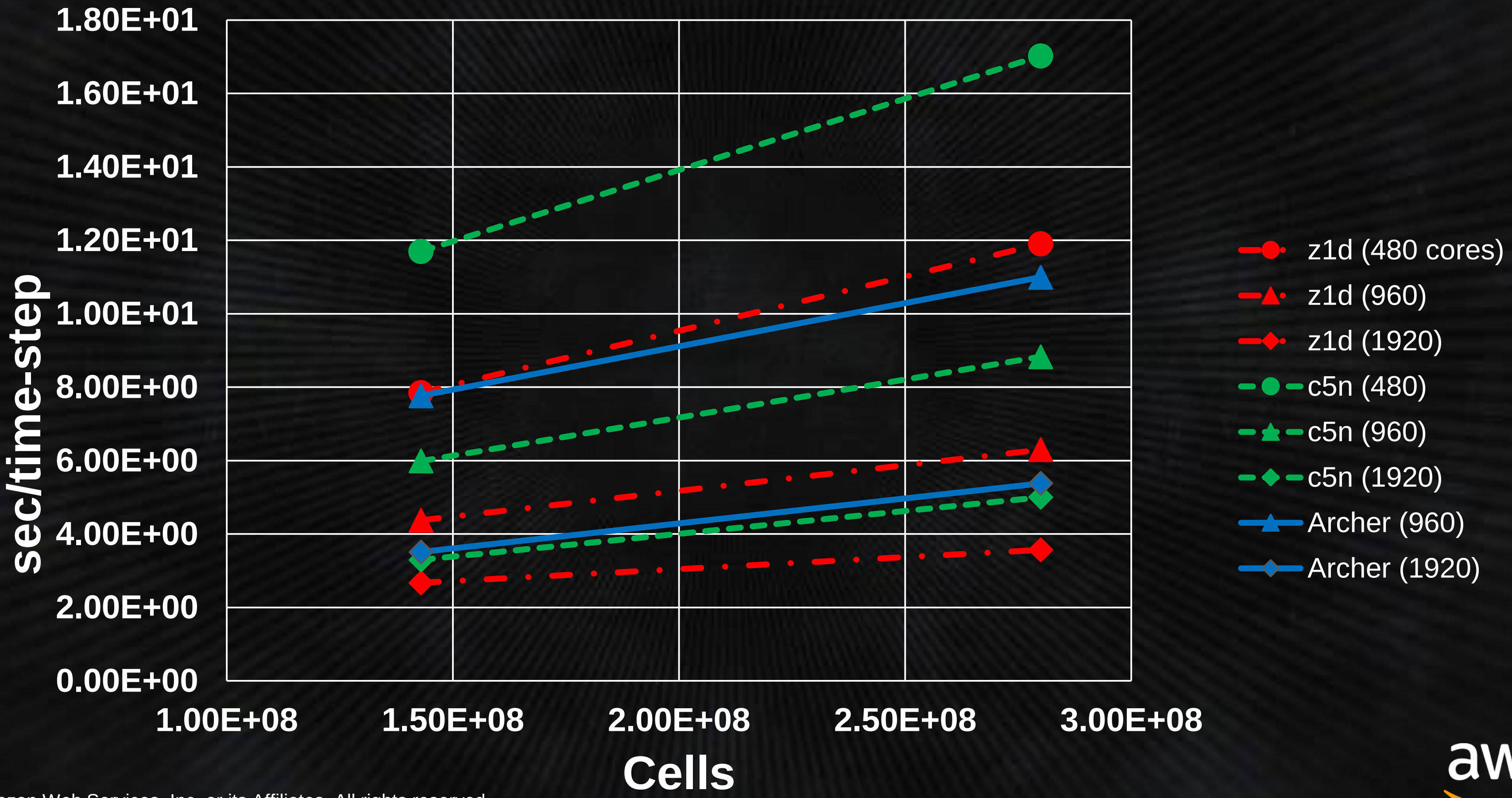
For example, scaling is measured at constant CFD mesh cells per core

Ideal: $\lim_{n \to \infty} speedup = 1 - p$

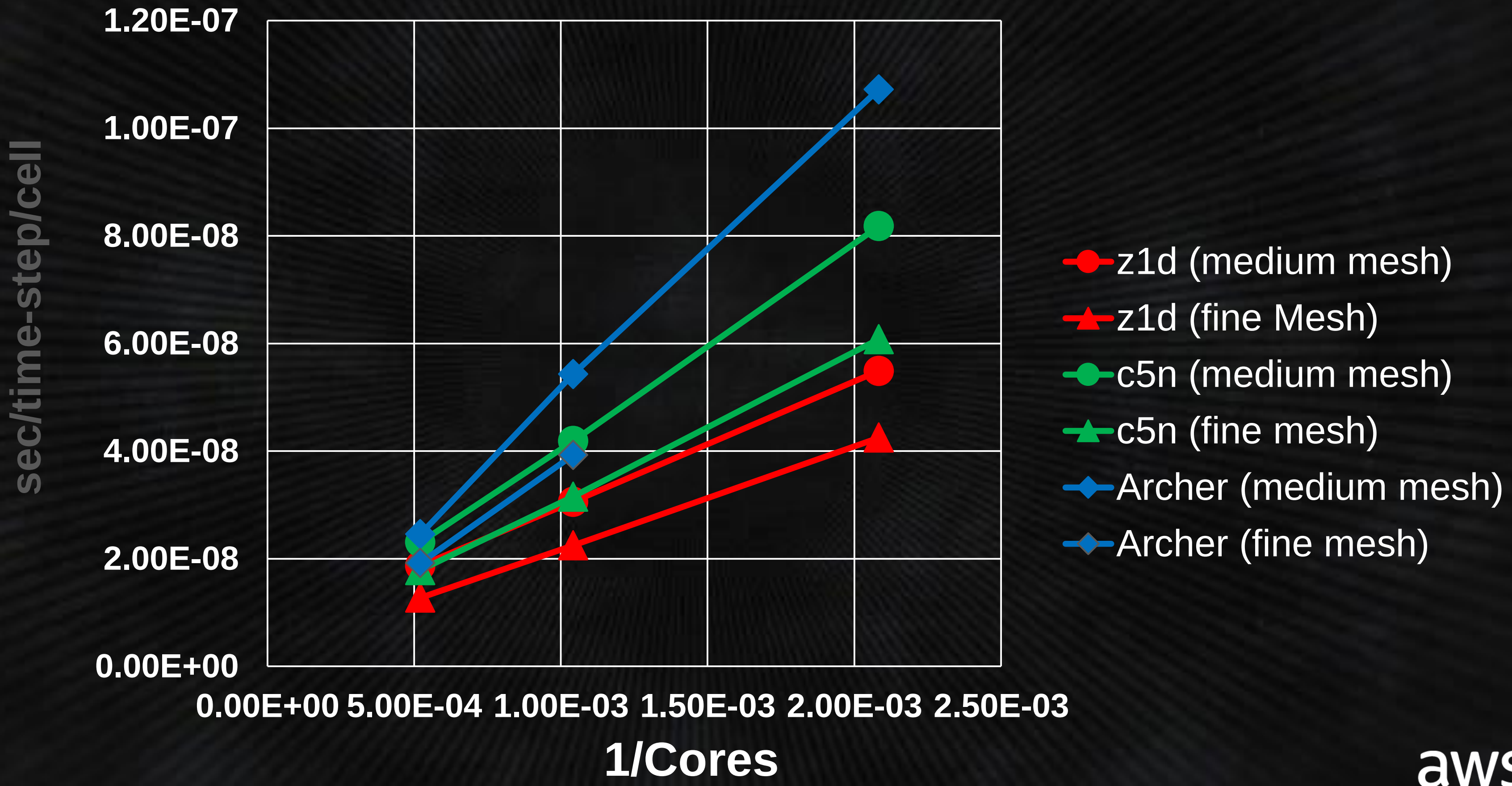
Gustafson Scaling: openFoam (pimpleFoam)



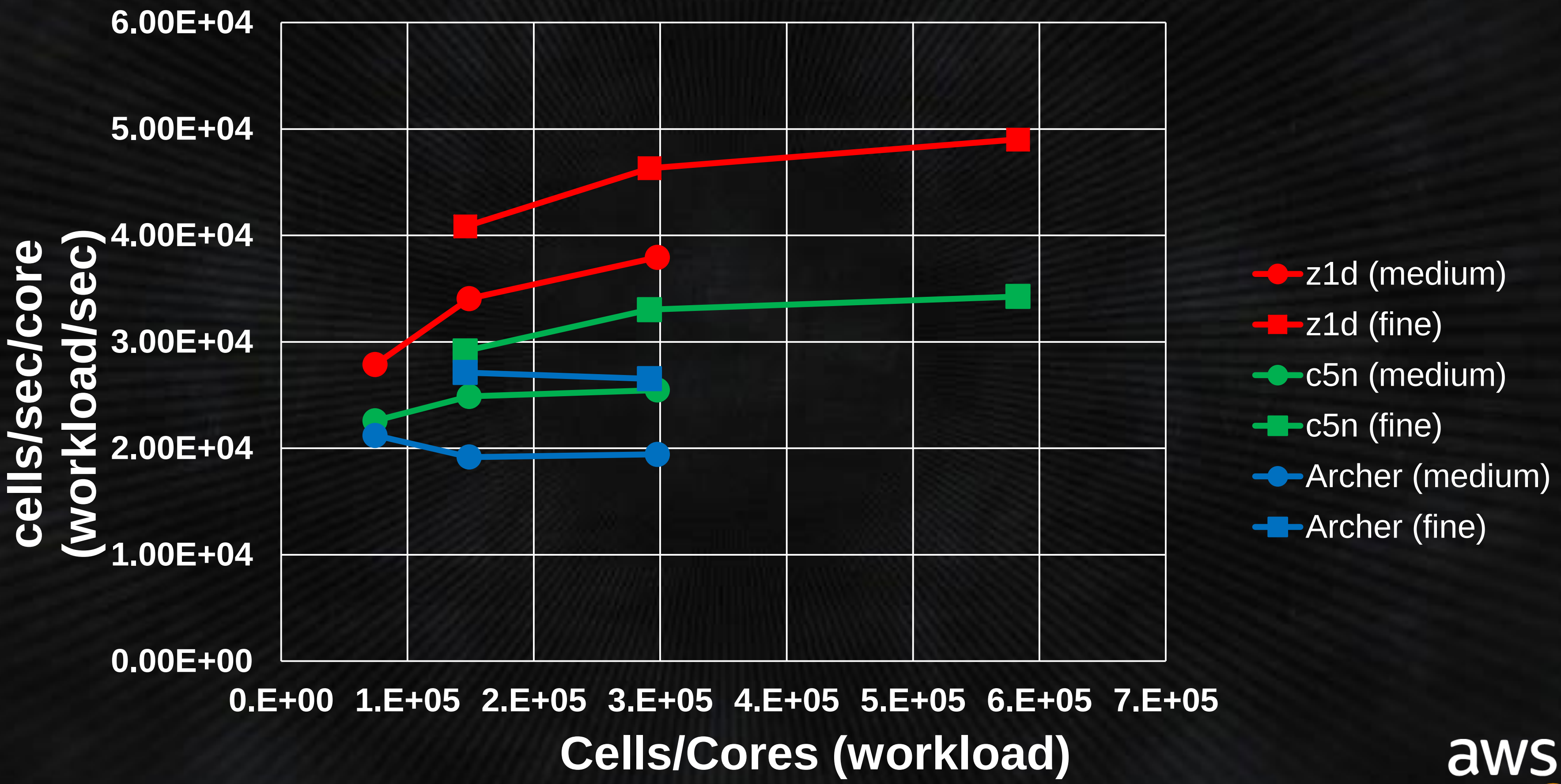
Gustafson Scaling: Alternate View



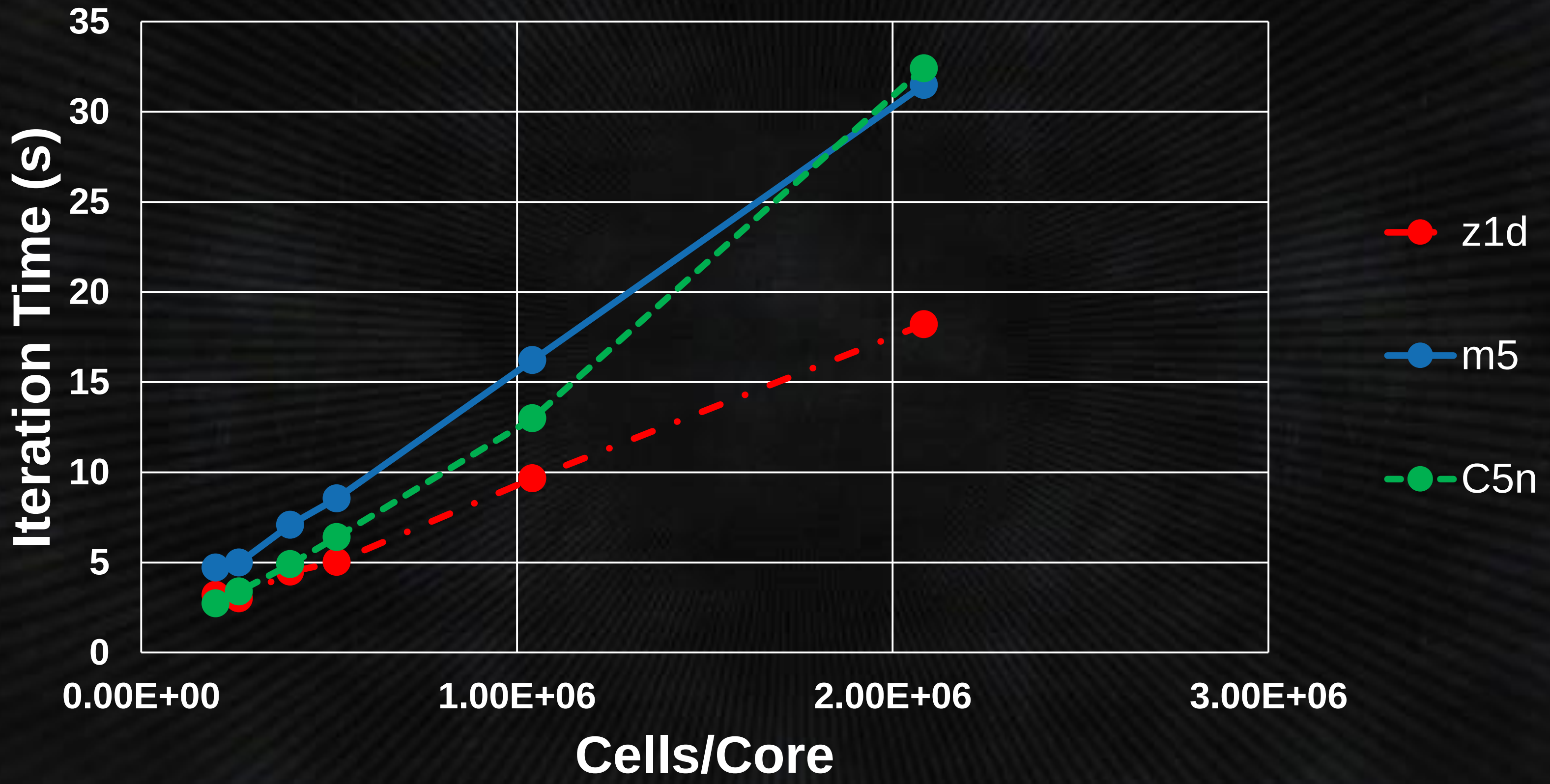
Gustafson: Speedup ∝ p/n



Gustafson: workload



400 Cell scaling



Obvious Question:

Why not include 400M cell case with others?

Answer:
They were done by two different people and two different workloads and, therefore, do not follow a consistent process

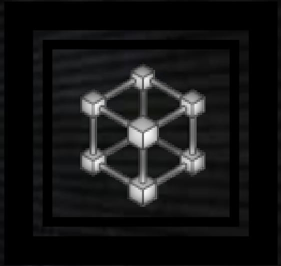
Latency

- Amdahl and Gustafson scaling both measure latency.   Latency includes:
  - Non parallel code
  - Memory latency
  - Cache latency
  - Network latency
  - Storage latency

Comments

- At high cell/core, or low core count, memory or cache more likely to slow processing (possibly see this in Archer)
- At low cell/core, or high core count, network latency more likely to slow processing (see in AWS-z1d and AWS-c5n)
- Total execution scaling based on processor workload (cells/core) is linear in all cases
- Slope of processor workload (cells/core) seems to follow network latency – faster network → steeper slope = better scaling
- Faster network does not mean faster execution time
- Coarser mesh has better scaling than finer mesh in all cases, but AWS-c5n has largest difference (best guess: PCG routines)

Discussion?

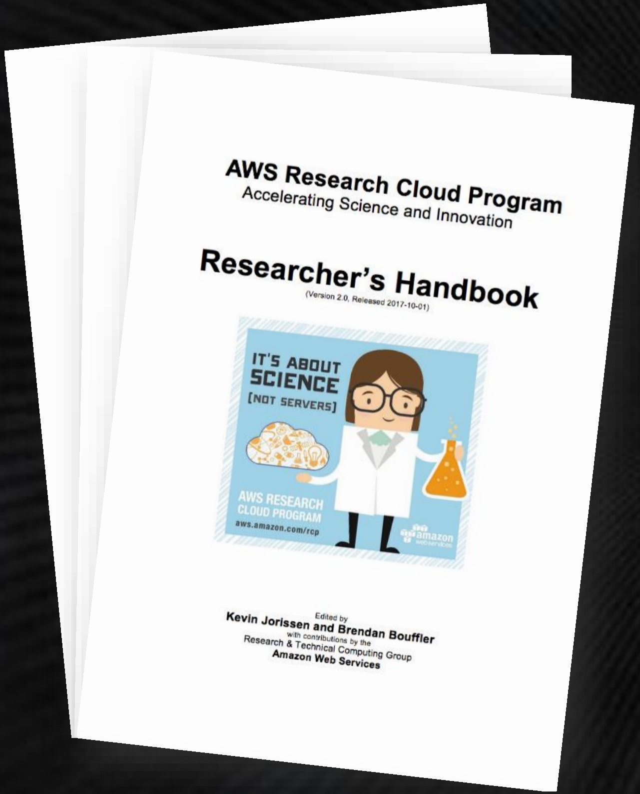


Acknowledgement

Special thanks to Dr. Neil Ashton of Oxford University for providing his openFoam benchmarking data for the 143M cell and 282M cell cases, and Stephen Sachs, of AWS for the 400M cell cases

AWS Researcher's Handbook

The 200-page "missing manual" for science in the cloud.



Written by Amazon's Research Computing community for scientists.

- Explains foundational concepts about how AWS can accelerate time-to-science in the cloud.
- Step-by-step best practices for securing your environment to ensure your research data is safe and your privacy is protected.
- Tools for budget management that will help you control your spending and limit costs (and preventing any over-runs).
- Catalogue of scientific solutions from partners chosen for their outstanding work with scientists.

aws.amazon.com/rcp

aws.amazon.com/rcp