



ORACLE[®]

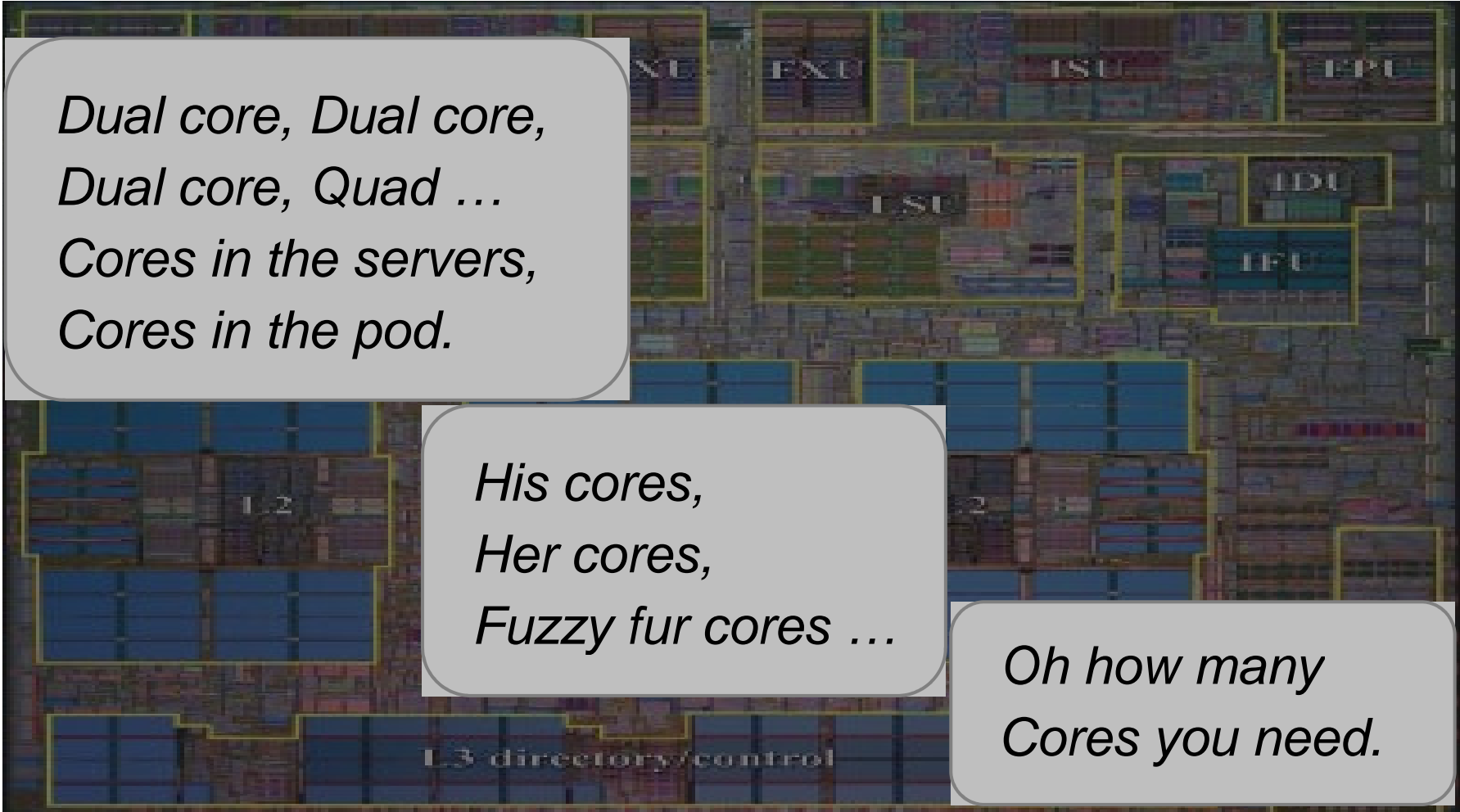
Traditional Programming Models:
Stone Knives and Bearskins in the Google Age

Cameron Purdy
Vice President, Development

Multi-server. Multi-processor.
Multi-core. All supported by a
fifty year old approach to
programming single-threaded
systems. It's time for a change.



Scores of Cores, Fours and Mores!



*Dual core, Dual core,
Dual core, Quad ...
Cores in the servers,
Cores in the pod.*

*His cores,
Her cores,
Fuzzy fur cores ...*

*Oh how many
Cores you need.*

Text adapted from "The Foot Book" by Dr. Seuss



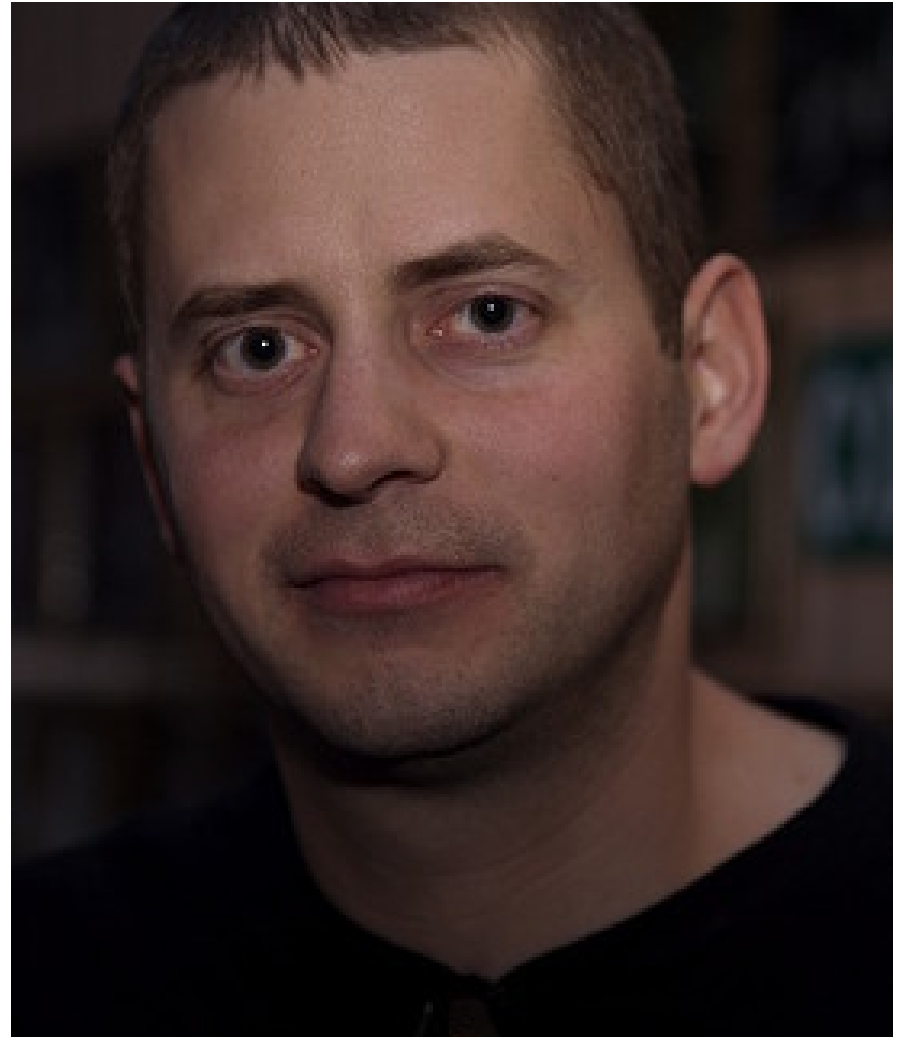
Introduction

Old ideas give way slowly; for they are more than abstract logical forms and categories, they are habits, predispositions, deeply ingrained attitudes of diversion and preference.

- John Dewey

About the Speaker

- Cameron Purdy is the Vice President of Development for Fusion Middleware, responsible for the Oracle Coherence Data Grid
- Founder and CEO of Tangosol, acquired by Oracle in 2007
- Working with Java since 1996, Java EE since 1999, and PowerPoint since 2007





Disclaimer

Religious application of principles learned from a presentation attended at an industry tradeshow is not a substitute for common sense.



The Challenge

There are two things in life
for which we are never truly
prepared: twins.

- Josh Billings

Challenge: Performance or Scalability?

- Why isn't the system fast enough? Was it fast enough with only one user? Then the challenge is scalability, **not** performance
- Queue theory almost always explains bad performance under load
 - *But why?*



Common Sense



- Horizontal scalability is the only answer
 - Too much work for one core? You need more than one core!
 - Too much work for one server? You need more than one server!
 - Complexity tends to increase exponentially in multi-threaded systems, and quadratically in distributed parallel systems



Defining the Challenge

The processor architecture and micro-architecture are undergoing a vigorous shaking-up. The major chip manufacturers have shifted their focus to "multi-core" processors The new focus is multiple cores with shared caches, providing increased concurrency instead of increased clock speed. As a challenge, software engineers can no longer rely on increasing clock speed to hide software bloat. Instead, they must learn to make effective use of increasing parallelism. This adaptation has never been easy.

- Xinmin Tian, Intel



Defining the Challenge

Now parallel processing capability is within the reach of every user. The question that naturally comes up is "Now that I have two cores, what can I do with them"? The short answer to this question is that you either:

1. Run more programs simultaneously (multi-tasking)
2. Parallelize your applications (multi-threading or parallel programming)

- Ron Wayne Green, Intel



Defining the Challenge

The change from single core to multi-core processors is expected to continue, taking us to many-core chips (64 processors) and beyond. Cores are more numerous, but not faster. They also may be less reliable. Chip-level parallelism raises important questions about architecture, software, algorithms, and applications. A chip-edge bandwidth crisis is looming, but new technologies may help us cope.

- Robert Schreiber, HP



Defining the Challenge

By 2021, there will be chips with 1024 cores on them. Is parallelism the tool that will make all these cores useful? John Hennessey has called it the biggest challenge Computer Science has every faced. He has credentials that might make you believe him. Allen says that it's also the best opportunity that Computer Science has to improve user productivity, application performance and system integrity.

- Phillip J. Windley

Fundamental Scalability Limiters

or “Why does queue theory explain bad performance?”

- Sequential high-latency operations
- Ordered operations (“dependencies”)
- Data “Hot Spots”



- Immediacy of reliable state
- Data consistency
- Durability of state change



Defining the Problem

The von Neumann architecture is a design model for a stored-program digital computer that uses a processing unit and a single separate storage structure to hold both instructions and data. It is named after the mathematician and early computer scientist John von Neumann. Such computers implement a universal Turing machine and have a **sequential architecture**.

- Wikipedia



Defining the Problem

Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the von Neumann bottleneck. Not only is this tube a literal bottleneck for the data traffic of a problem, but, more importantly, it is an intellectual bottleneck that has kept us tied to word-at-a-time thinking instead of encouraging us to think in terms of the larger conceptual units of the task at hand. Thus programming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself, but where to find it.

- John Backus, 1977 ACM Turing award lecture



Messaging Concepts

- Delivery Guarantees
 - None (“Unreliable”)
 - Reliable
 - Guaranteed (Durable)
- Ordering Guarantees
 - None
 - Approximate Ordering
 - Point-to-Point Ordering
 - Global Ordering
- Latency
 - FIFO
 - Prioritized
 - Bounded
- Processing Guarantees
 - Best Effort
 - At Most Once
 - At Least Once
 - Once And Only Once

Programming Models vs Messaging

- Efficient “von Neumann” Programming Languages
 - Single threaded
 - Reliable
 - In order
 - Zero latency invocation
 - Once and only once processing



- Messaging makes you pay
 - Ordering limits scalability
 - Reliability adds costs
 - Introduces latency
 - Once-and-only-once is hard



The Solution

A bend in the road is not
the end of the road...
unless you fail to make the
turn.

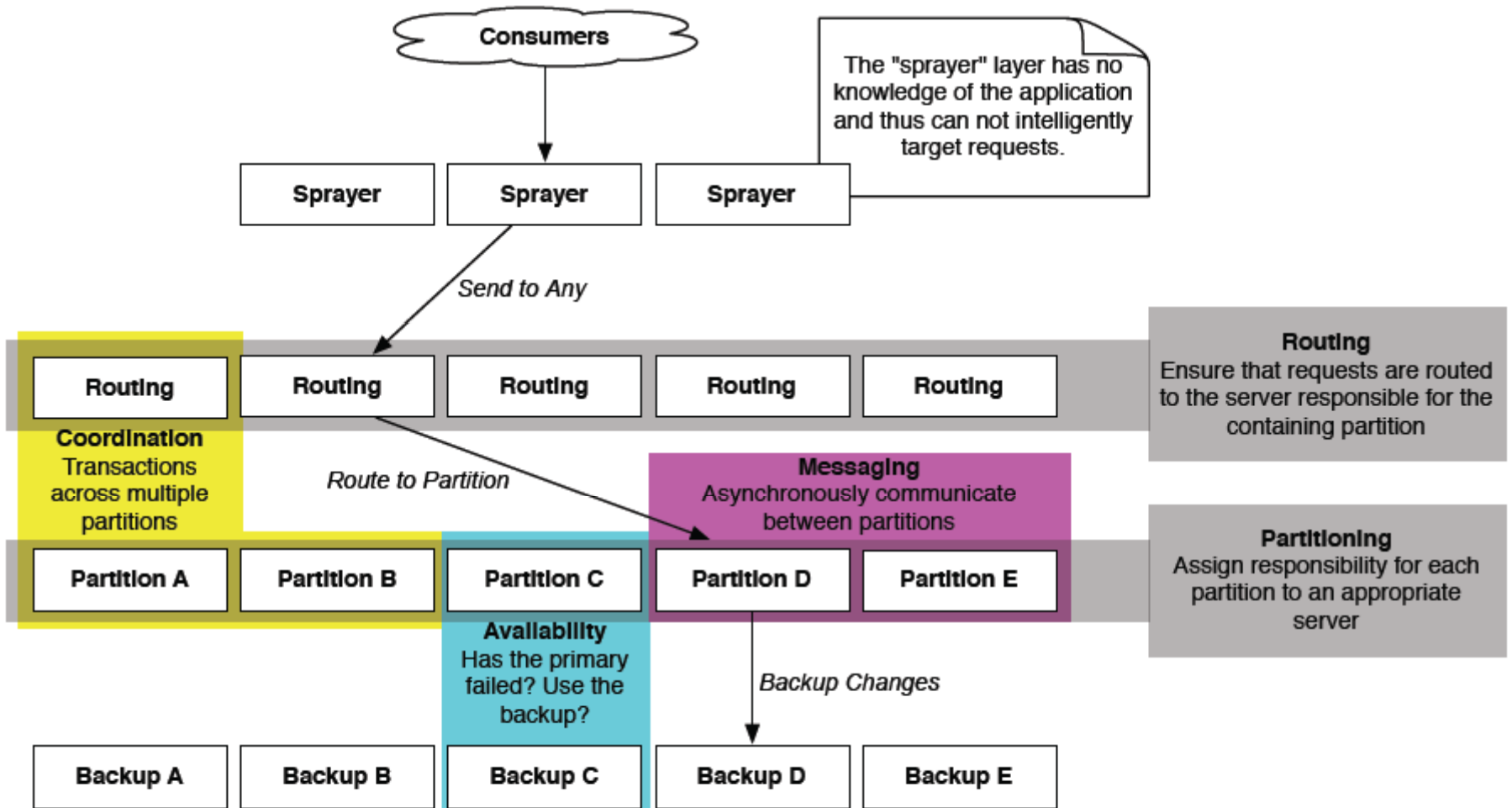
- *Anonymous*

Scalability: How to Deal with Load

- There are only so many ways to deal with load
 - Load Balancing: An arbitrary spreading of load (spraying)
 - Partitioning: Slicing a stateful domain and routing work accordingly
 - Relax data consistency requirements
 - Parallelization: Concurrent processing by spreading horizontally
 - Queue: Collect work that can be performed more efficiently in bulk



The Five Patterns of Stateful Scale-Out



Partition for Scale

- Partitioning is the only way to provide significant scale in a stateful system
- Requires identity+granularity
- Introduces challenges for distributed read consistency
- Joins become inefficient
- Even with custom affinity of data within a distributed system, complete data locality is simply impossible for some applications



Relaxing Data Consistency

- CAP Theorem: Consistency, Availability, Partition-tolerance – choose any two
- Caching: Relaxing read consistency
 - Allows the system to serve data in a more efficient and scalable fashion
- Eventual Consistency: Relaxing write consistency
 - Allows the system to accept new data in a more scalable fashion



Queue

- Data access
 - Read coalescing
 - Write coalescing
 - Write batching
- Event Driven Architectures
 - Enqueue the next set of operations while the current set of operations are executing, e.g. matching engines
 - Commit state changes *en masse*
- Dynamic: Queue more aggressively as load negatively affects latency



Parallelize

- Enabled by partitioning
- Eliminate sequential operations when possible
 - Change *for-each* thinking to *against-each*
 - Convert the block of the for loop into a command or functor
- Examples
 - Map/Reduce (e.g. Hadoop)
 - Data Grid
- Languages
 - Erlang, Scala





Defining the Solution

Event driven architectures support the decomposition of complex transactions into series of idempotent operations, each of which raise events that naturally trigger the next operation in the series in a predictable and reliable manner. Localizing these operations within a scale-out environment provides in-memory execution speed, and data affinity enables the batching of those operations. With smaller operations that can be processed in parallel, the result is an approach that can make very effective use of large-memory, multi-CPU/multi-core systems in scale-out environments to drive transaction volumes that are orders of magnitude greater than what can be accomplished with traditional TP approaches.



For More Information

<http://search.oracle.com>



or

<http://www.oracle.com/technology/products/coherence/index.html>



ORACLE[®]

Traditional Programming Models:
Stone Knives and Bearskins in the Google Age

Cameron Purdy
Vice President, Development