A CASE FOR INTELLIGENT (CO-)DESIGN

Sean Treichler, April 25, 2017

NVIDIA

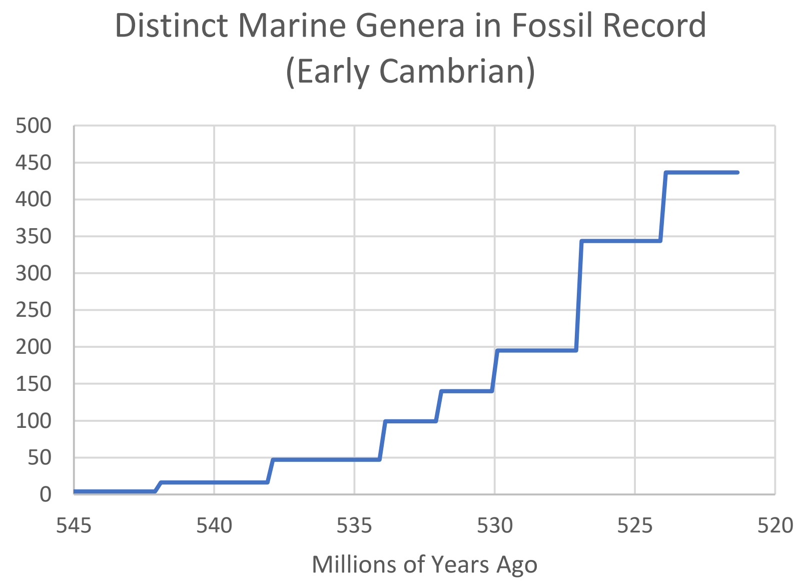
EARLY CAMBRIAN PERIOD

Period of hyper-inflationary growth in biological diversity

Most modern animal phyla can be traced to Cambrian

Believed to have been driven by random mutation and natural selection

Enabler is uncertain: Rapid increase in resources?



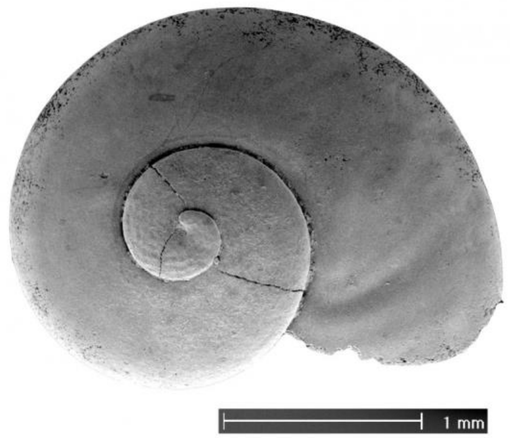
Source: Robert A. Rohde and Richard A. Muller
Nature **434**, 208-210(10 March 2005)

MANY NICHES

Studies of biological ecosystems teach us the efficiency of diversity

Different niches require different abilities (e.g. photosynthesis, mobility)

Adaptations for one niche generally reduce competitiveness in others



http://geokogud.info/specimen_image/46431

https://en.wikipedia.org/wiki/Wiwaxia#/media/File:Wiwaxia_5C_crop.jpg

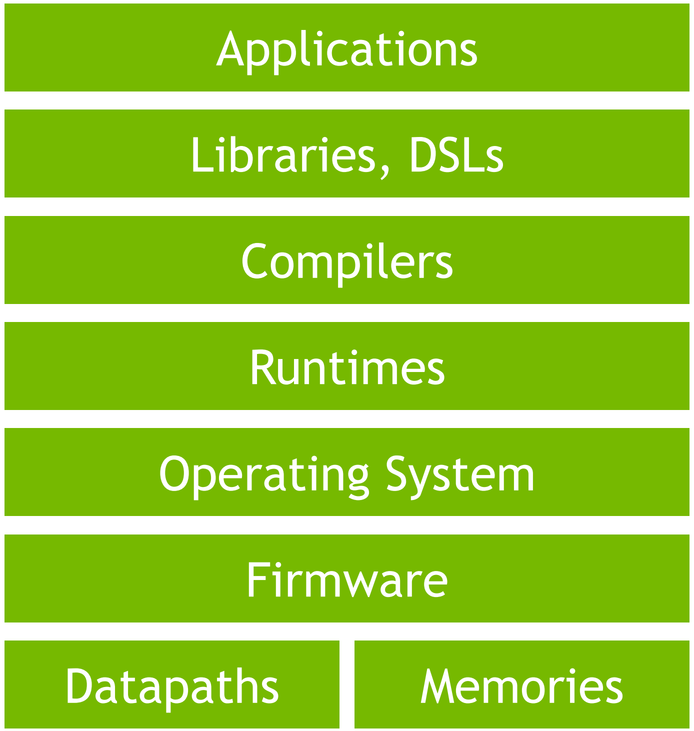
COMPLEX ORGANISMS

Our niches are applications or groups of related applications

Our organisms are “computing systems” made up of software and hardware components

Heterogeneity exists in (at least) three forms:

- Use of different components in single application
- Use of different components in different applications
- Changes in the quantitative mix of the same components in different applications



NATURAL SELECTION

Simple and “elegant”, but slow and incredibly wasteful

Not really “survival of the fittest”, rather “elimination of the inferior”

Significant resources and effort go into each eliminated individual

Large amount of randomness (“noise”) in the process

Most eliminations do not measurably improve the population

Most mutations change only a single thing

IRREDUCIBLE COMPLEXITY

"A single system which is composed of several interacting parts that contribute to the basic function, and where the removal of any one of the parts causes the system to effectively cease functioning." (Michael Behe, *Darwin's Black Box*)

Problem of local minima in non-convex optimization

Nearly all random perturbations yield inferior results

Inefficiency is a serious problem for machine-driven optimization

Inefficiency is a deal-breaker for human-driven optimization

CO-DESIGN

Need to be able to change multiple components at the same time

Starting from scratch is intractable, so focus on “irreducible cores”

Come up with a “language” to talk about the behavior of that core

Language should be descriptive (what) rather than prescriptive (how)

Identify one or (likely) more right answers for various use cases

Decide which components are best suited to implement desired solution

EXAMPLES OF CO-DESIGN

None of these are “paid endorsements”

Any inaccuracies are my fault

Not an exhaustive list

LEGION

Bauer, Treichler, and Aiken, SC ‘12

Programming model for performance portability on heterogenous, distributed systems

Based on a hierarchical decomposition of code (tasks) **and** data (regions)

Explicit mapping of application onto target machine(s)

Separation of policy and mechanism

Composability of software libraries, domain-specific languages

HIHAT

https://wiki.modelado.org/Hierarchical_Heterogeneous_Asynchronous_Tasking

Exploring common API for applications/runtimes that desire portable task-parallelism

Initiated by NVIDIA, hosted by Modelado, open to all

"Common layer" – What are the fundamental primitives for initiating computation and communication?

Very early stages – gathering usage models and requirements

First mini-summit: May 9 in San Jose (near GTC)

LLVM

Lattner and Adve, 2002

Modular and reusable compiler and toolchain technologies

Key enabler of many efforts in program transformation, domain-specific languages

Supports JIT (Just In Time) compilation

What information can be used to generate optimized machine code?

What trade-offs are possible in performance vs. effort?

TAPIR

Schardl, Moses, and Leiserson, PPoPP '17

Problem: Most compilers are unable to reason about inter-thread behavior

Extends LLVM IR with instructions that capture Cilk-style (i.e. fork-join) parallelism

Enables simple optimizations across threads (e.g. sync elimination, inlining)

Many existing LLVM optimizations work with no/minimal changes

Should be extensible to more general forms of parallelism

COMMUNICATION-AVOIDING ALGORITHMS

Demmel et al, 2011-present

Data movement is increasingly a performance bottleneck

Theoretical bounds for various algorithms drive optimization, enable balanced architectures

Recent work allows automatic discovery of optimal algorithm in many cases

Exploration of algorithmic alternatives allows applications (or runtimes?) to make communication/computation trade-offs

DUNE

Belay et al, OSDI ‘12

Processors/systems contain features designed for multi-user systems

How can these be of use to a single application?

Privilege levels: sandboxing of web browser (or stochastic superoptimizer?)

Virtual memory: better garbage collectors (or background checkpointing?)

Could future systems treat some/all of OS as part of “application”?

INVESTING IN THE FUTURE

Co-design isn’t free

Reduces the dimensionality of our design space

Allows solutions for one case to be applied to others

Better human understanding of the problem allows us to make intuitive leaps

LEARNING FROM THE PAST

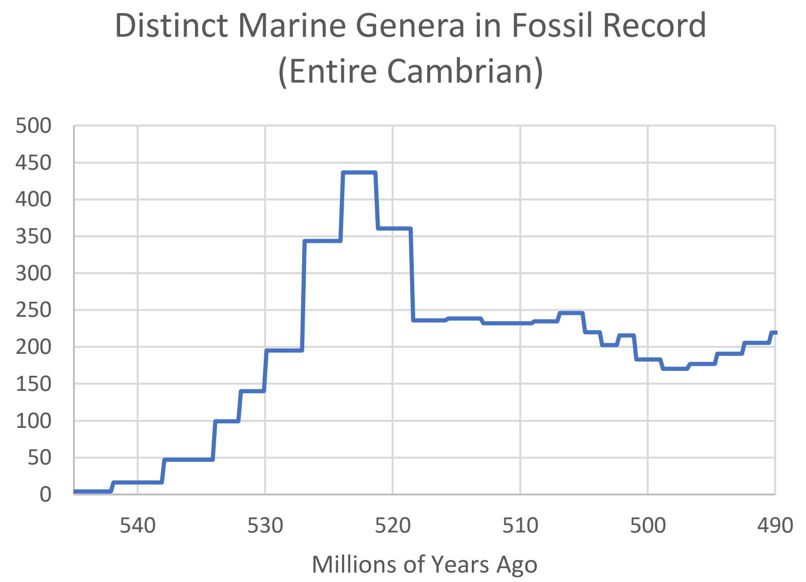
No party lasts forever

Consolidation as resources change, niches merge/vanish

Natural selection ignores “potential” or “aesthetics” or “big picture”

Knowledge/effort of extinct species are lost

Co-design allows learning from negative results, reuse of positive results



Source: Robert A. Rohde and Richard A. Muller
Nature **434**, 208-210(10 March 2005)

NVIDIA