LA-UR- 11-02499

Approved for public release; distribution is unlimited.

Title:	Are We Exploring the Right Enabling Technologies to Support End Applications in the Push to Exascale? (U)
Author(s):	Robert B Webster, LANL
Intended for:	Distribution as required

Los Alamos
NATIONAL LABORATORY
EST. 1943

Form 836 (7/06)

Are We Exploring the Right Enabling Technologies to Support End Applications in the Push to Exascale?

Robert Webster, ***Los Alamos National Laboratory***

Abstract:

Modern applications in use within the Defense Programs mission of NNSA can greatly benefit from the development and use of exascale-class computing. To take advantage of such computing, the applications will have to evolve to fit the modern architectures required to achieve such a scale in computing. Because of the long development time for these applications, much discussion and thought is being given to programming models, and the role that co-design could have in developing a hardware and software model that minimizes the cost of moving applications between changing and potentially divergent architectures, hiding data movement, exposing (or hiding) power management, and the like. Perhaps instead, the community should target the requirement for an application rather than the application itself, and thus target reducing the time to develop an application as opposed to the cost of maintaining and moving the application between architectures. Some motivational examples from the past are discussed, as well as the role of continued application development in the path to exascale.

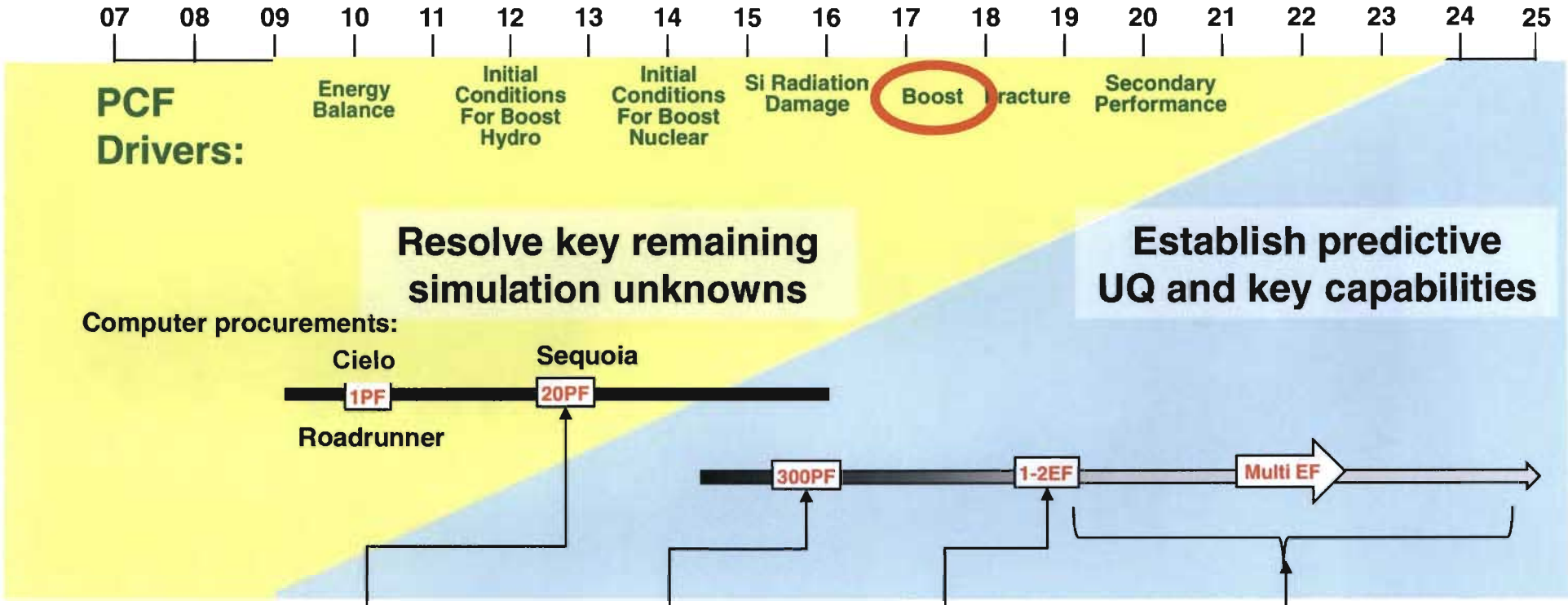
A large "fixed cost" of personnel for NNSA Apps opens opportunities for bridging architectures

- The ASC right-size effort establishes the scale of effort necessary to maintain nuclear weapon simulation competence.
- The manner in which present day decisions are made limits the number and type of options available to stockpile stewardship
  - The available options are nearly exhausted
  - Desirable options require improved physics

    We don't know what the required physics improvement is!!!
  - At present, it takes about a decade to build and deploy a weapon simulation tool.
- We need to iterate fundamental physics and modeling in our codes more quickly than is possible at present.
  - An appropriate target is 4 years to develop a weapon performance app.
  - It is necessary that the app does not significantly lag the hardware.

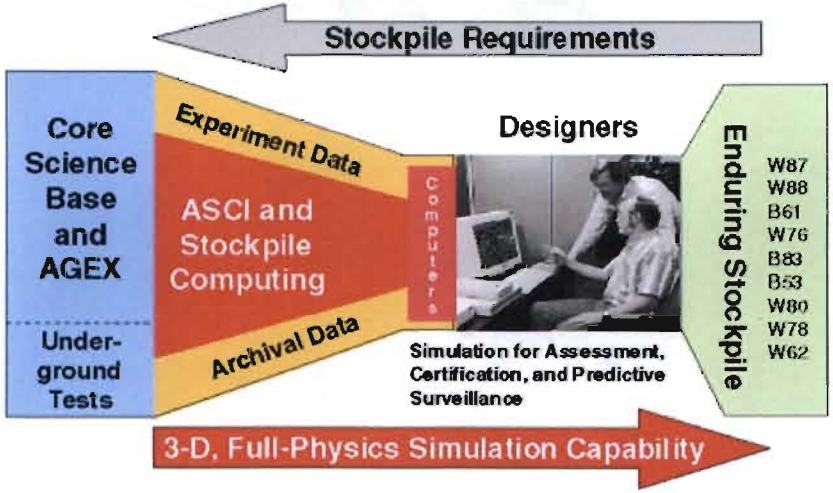
For NNSA Defense programs the cost of rewriting the applications is a secondary issue, the time to rewrite the applications is paramount!

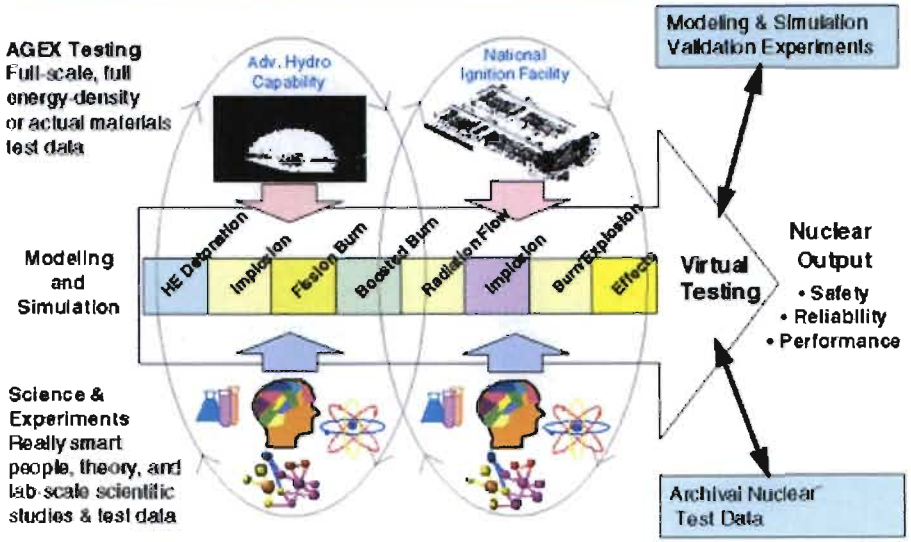
Nuclear Weapons requirements drive the need for greater understanding, and 3D predictions with quantified uncertainty



We have learned a lot since the Accelerated Strategic Computing Initiative (circa1996)

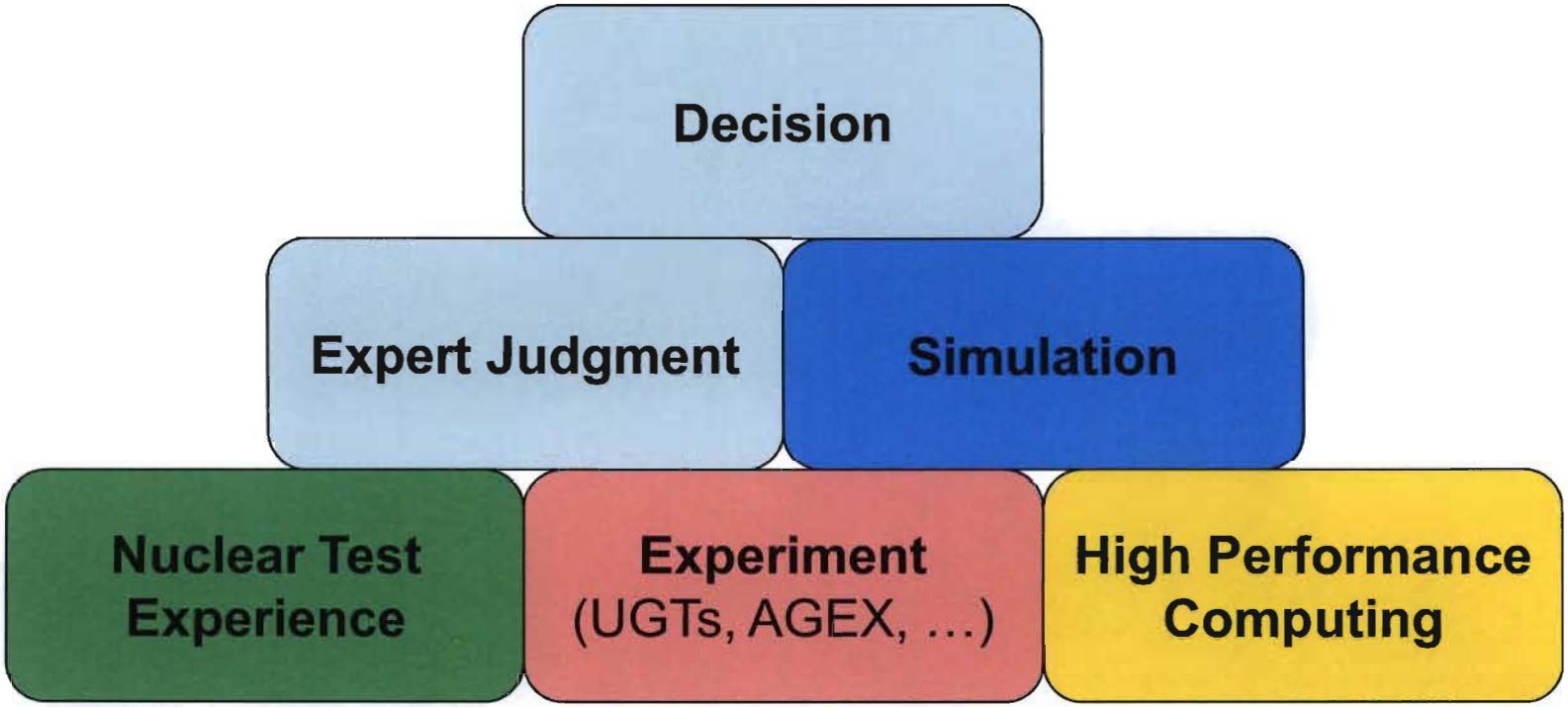
- Petascale allows 3D simulation or perhaps full physics, but not both
- Simply scaling up our apps does not solve Boost





- The basic philosophy is still being pursued.
- Changes in computing use cases resolved some of the worries of the day:
  - Checkpoint/Restart
  - Graphics demands

Decisions regarding nuclear weapons are still made by a combination of expert judgment and simulation



Today:

Certify/assess with scale dependent physics.

=> Requires proximity to Nuclear Tests (ok for limited SFI's, LEP's, etc.)

"Expert Judgment" informed by NTS experience

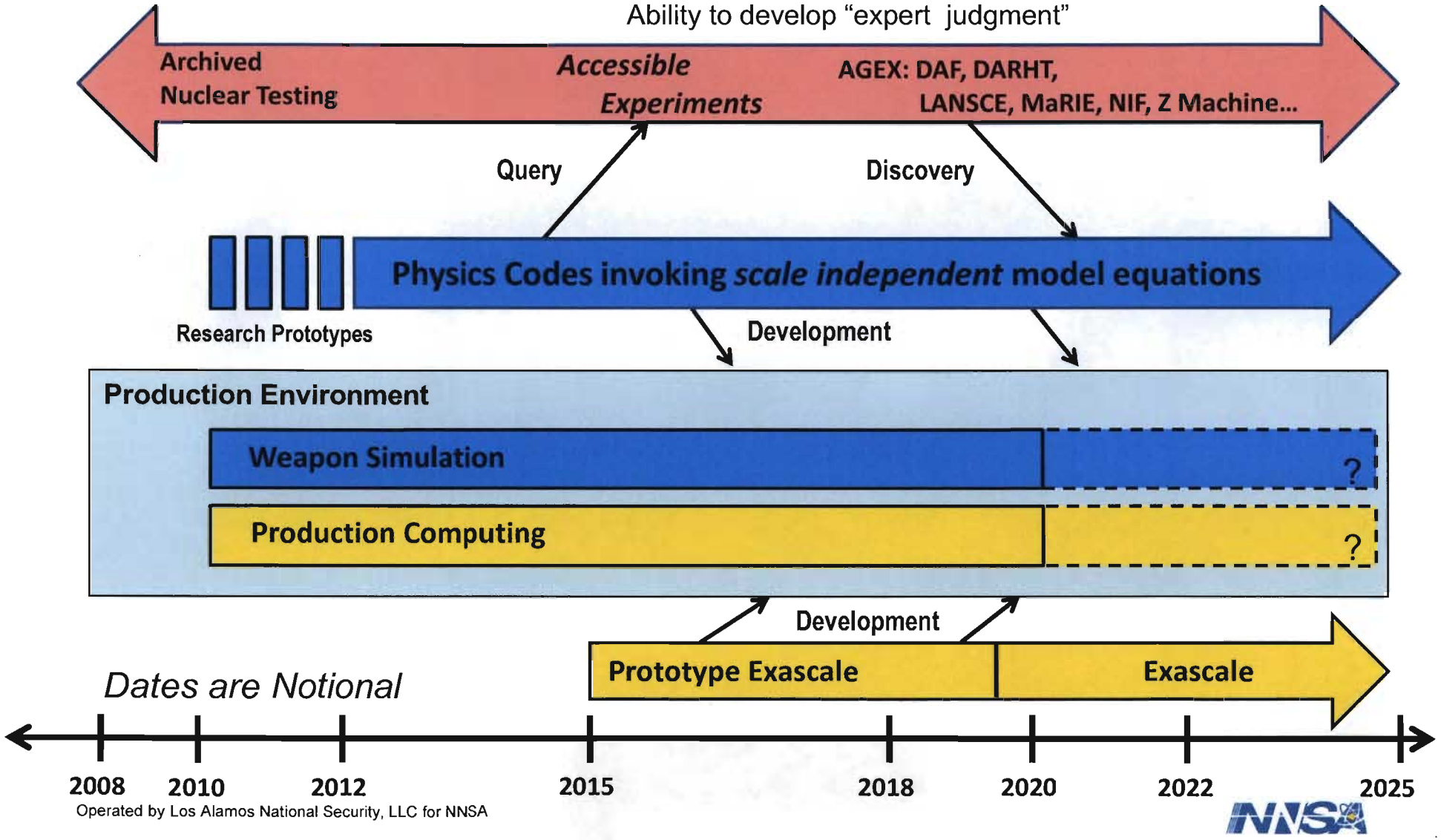
Tomorrow:

Codes have scale independent physics models

=> Expanded domain of calculation "validity"

"Expert Judgment" developed from AGEX + New Codes + Nuclear Test History

Goal — to enable decisions based upon reliable predictions in experimentally inaccessible regimes!



Present Day Algorithms scale very poorly => Not suitable for our "physics code"

- **Suppose today's algorithm as applied at exascale has:**
  - Weak Scaling
  - 1000x performance
  - 1000x memory ;)
  - I/O that keeps up…
- **Existing Navier Stokes Solvers will achieve 5.6 times today's resolution in 3D**
  - That does cross an interesting physical scale length, but…
  - The other physics in our codes may not be valid at that scale,
  - We go from resolving 2 decades of inertial range to 2 ½ .
  - (Non-hydro physics can scale even more poorly!)
- **We are studying new algorithms that can better take advantage of the scale of computation, such as the High Order/Low Order radiation transport model in development at LANL**
  - May lose determinism
  - Checkpoint requirements can become much smaller

Bottom line, only marginal discovery remains with the brute force approach

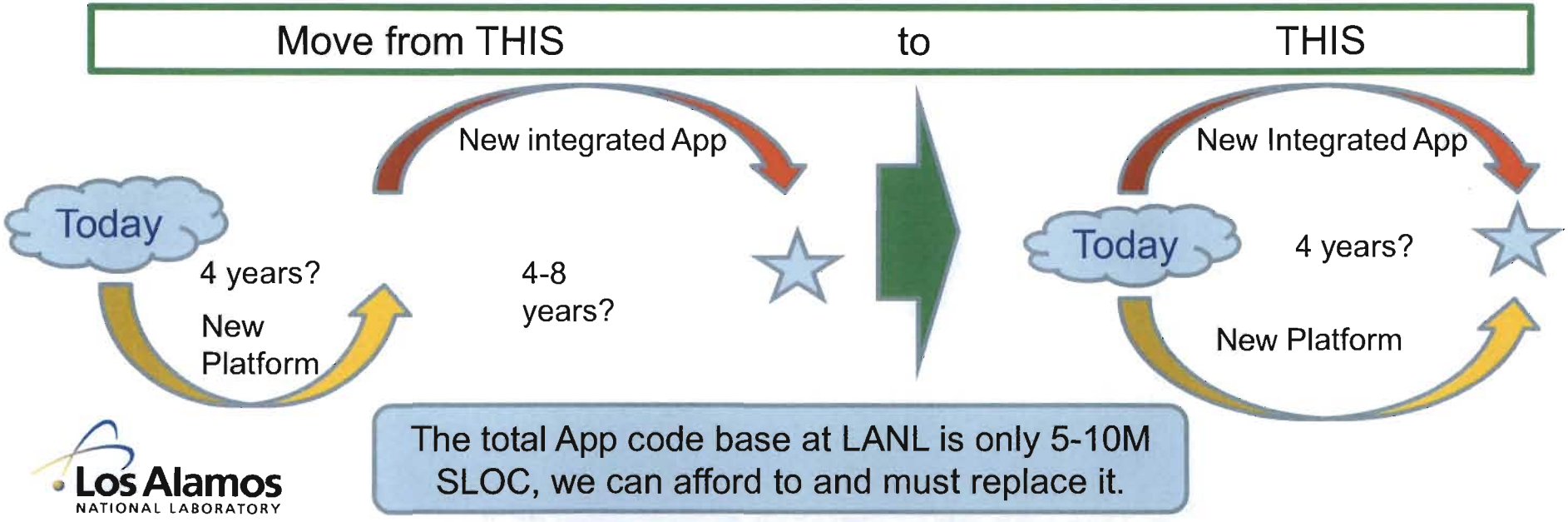
We use HPC to perform scientific discovery through complex deductions

- **"In July 2003, Raymond Orbach, Director of the DOE Office of Science, testified before the U.S. House of Representatives Committee on Science. He said**
  - **The tools for scientific discovery have changed. Previously, science had been limited to experiment and theory as the two pillars for investigation of the laws of nature. With the advent of what many refer to as "Ultra-Scale" computation, a third pillar, simulation, has been added to the foundation of scientific discovery."**
- **This type of scientific discovery REQUIRES re-development of the high end applications**
  - ***Such re-development constitutes the hypothesis in play***
  - ***Rapid re-development is the key to scientific leadership***
  - ***Rapid re-development makes evolution look like revolution***

Quote from GETTING UP TO SPEED:
THE FUTURE OF SUPERCOMPUTING
THE NATIONAL ACADEMIES PRESS - 2005

We need a revolution in app design, lots of possibilities exist

- **Non-deterministic computing**
  - Smaller checkpoints and graphics files
- **New algorithms**
  - Can also impact the time integration scheme
- **Change of the primitive state variables stored**
  - Typically this is a blank sheet startup
  - Potential impact on memory access coherence



The total App code base at LANL is only 5-10M SLOC, we can afford to and must replace it.

But, the community seems to be driven towards revolutionary CS

- **Quoting John West from "There is a right way, and a wrong way, to exascale | insideHPC.com"**

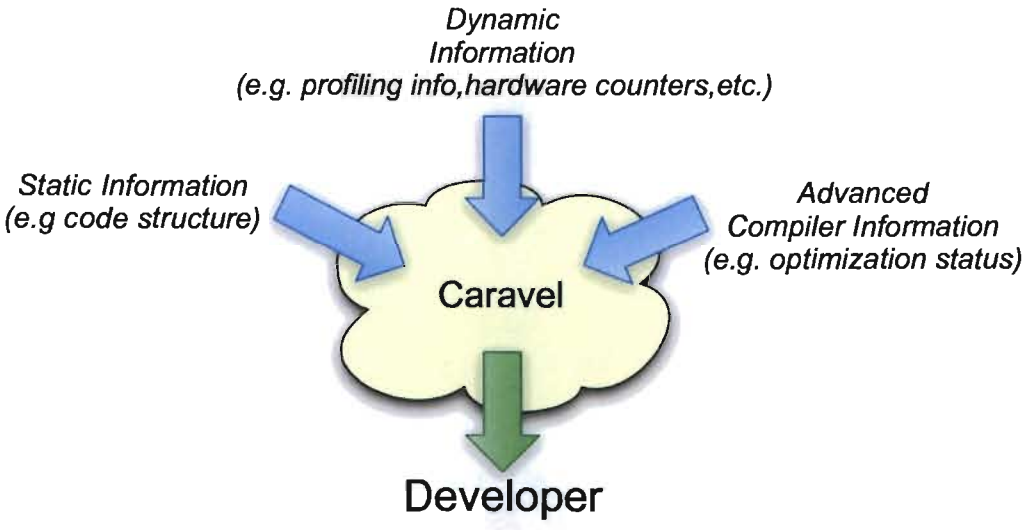
  " [snip] My own reason for agreeing with this point of view is that while, yes, we can build petascale machines, we are getting between 1% and 5% of peak on general applications. This is what an evolutionary model gets you. We are well past the point when a flop is worth more than an hour of application developer's time. We need to encourage the development of integrated hardware/software systems that help programmers write correct, large scale applications that get 15, 20, or even 30% of peak performance. To mangle Hamming, the purpose of supercomputing is discovery, not FLOPS."

- **There is a pervasive tendency to fear the cost of application rewrite, or wasted flop efficiency**
- **To little attention to the role of memory in the applications.**

If our "apps" programmers could get significant flops from a toaster they'd program it. The stockpile was designed with explicit memory management, overlays and significant coding in machine and assembly language.

Caravel: Program-structure-based exploration and performance analysis

- Compilers promote a "fire-and-forget" mentality
  - Details left unexplained – a black box, useful information lost
- Difficult to wade through a complex code and its characteristics (compiler feedback, performance, source, etc.)
- Goal: Give developers better capabilities



Support code-centric queries
Performance-centric details
Correlate cross-tool results
Embedded feedback

Concepts developed in the context of xRAGE, an ASC integrated code

So what do the Apps need?

- **Yes we need most of the current efforts:**
  - Domain Specific Languages (like Liszt) look promising
  - RAS will be needed regardless of programming model
  - Resiliency will be easier if hidden
  - Program analysis tools
  - Programming models for hierarchies
- **Not clear that :**
  - We need languages to hide architectures
  - Checkpointing is a show stopper
- **What do we need that isn't evident at the Apps end yet?**
  - Simulators so that we can design to the future architecture
  - Memory compression schemes
  - Variable precision arithmetic
  - Debugging tools for non-deterministic computation
  - Test and build systems that span sites, architectures, and scales

Memory is king for present approaches to discovery for NNSA/DP

A large "fixed cost" of personnel for NNSA Apps opens opportunities for bridging architectures

- The ASC right-size effort establishes the scale of effort necessary to maintain nuclear weapon simulation competence.
- The manner in which present day decisions are made limits the number and type of options available to stockpile stewardship
  - The available options are nearly exhausted
  - Desirable options require improved physics

    We don't know what the required physics improvement is!!!
  - At present, it takes about a decade to build and deploy a weapon simulation tool.
- We need to iterate fundamental physics and modeling in our codes more quickly than is possible at present.
  - An appropriate target is 4 years to develop a weapon performance app.
  - It is necessary that the app does not significantly lag the hardware.

For NNSA Defense programs the cost of rewriting the applications is a secondary issue, the time to rewrite the applications is paramount!