

Nuclear Weapons Design Codes: A 50 Year Perspective

LA-UR-04-2501



NTS Grable
25 May 1953
15 kT Airburst

Bill Chandler -- chandler@lanl.gov
Mike Berry -- mberry@lanl.gov

X-3 Group
Los Alamos National Laboratory



Salishan Conference on High Speed Computing - 2004

04/20/04 - 1



Abstract

TITLE: Nuclear Weapons Design Codes: A 50 Year Perspective

From VonNeumann using the Eniac to today's ASCI code developers using tens of teraflops on the ASCI computers; enormous progress has been made in modeling and understanding the physics and performance of nuclear explosives. And yet – we still have a long way to go if we truly expect to use our codes for predictive capability instead of relying on nuclear tests. This talk will review the significant hardware and software developments of the past decades and its impact on the development of nuclear design codes. There have been successes and failures, and despite the billions of dollars spent, significant challenges remain. How many flops and how many processors will it take to answer the questions? This talk will present the author's perspective on the problem, dig into a little history and try to speculate on the future.

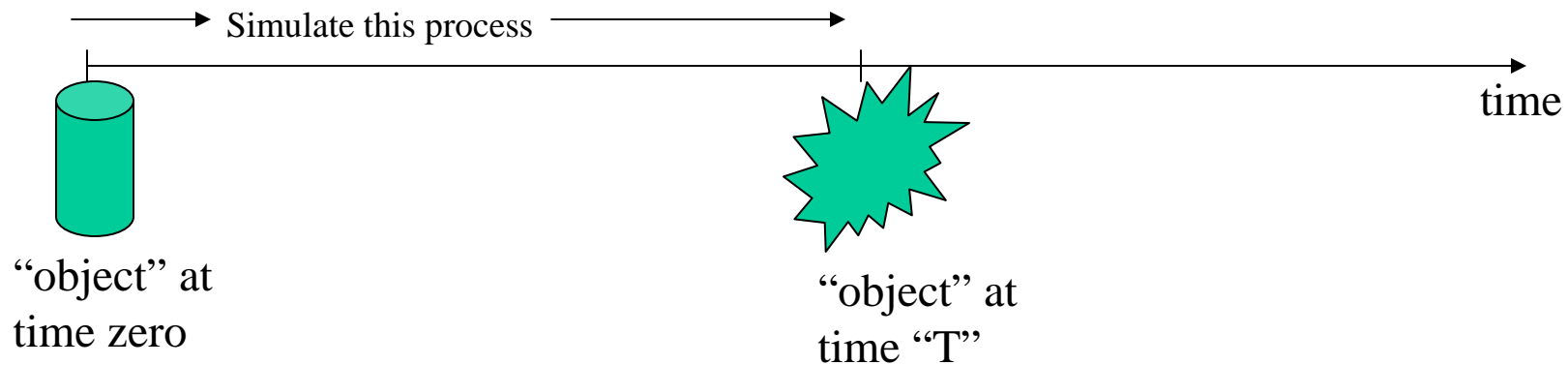


Outline

- What is **THE** application?
 - Answer: weapons design codes
- The Challenges:
 - Physics
 - Numerics
 - Computational
- Some History
- The Future



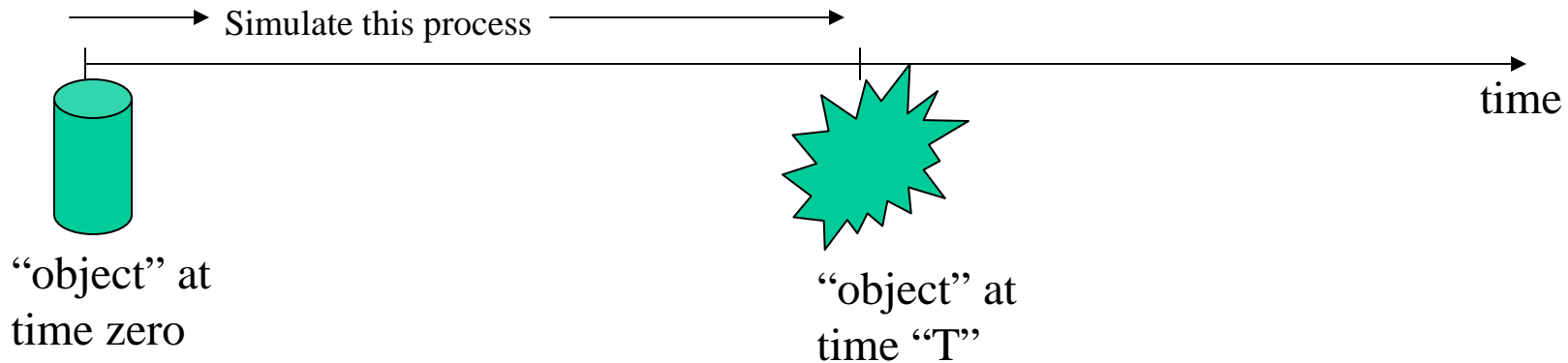
So...what is a design code?



- Model the evolution of the "object" from $t=0$ to $t=T$. Determine nuclear energy created and characteristics of the energy.
- Solve time dependent, non-linear, highly coupled, multi-physics, multi-dimensional differential equations describing "object" between $t=0$ and $t=T$.
- And get it right...and the stakes are HUGE...

And....you currently cannot test to see if you've got it right!

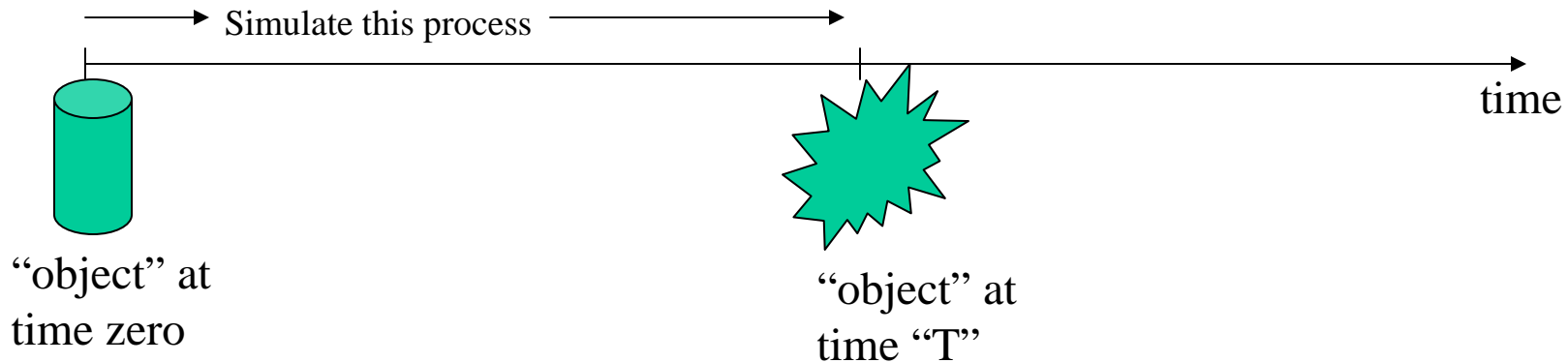
Weapons Simulation: The Physics Challenge



Physics considerations

1. Assembly of critical mass - hydrodynamics
2. Materials (nuclear fuels) fission and/or fusion - (they “burn”)
3. Materials go from room temperature (solids) to extremely high temps...rapidly
4. Interaction of nuclear products and energy with materials allows “object” to burn efficiently (or to not burn efficiently - ie., it doesn't work)
5. Fundamental physical data over temperature and density space (cross-sections, material properties, etc.) is an enormous challenge
6. Effects of aging (What are the physical properties of the parts after 50+ years?)

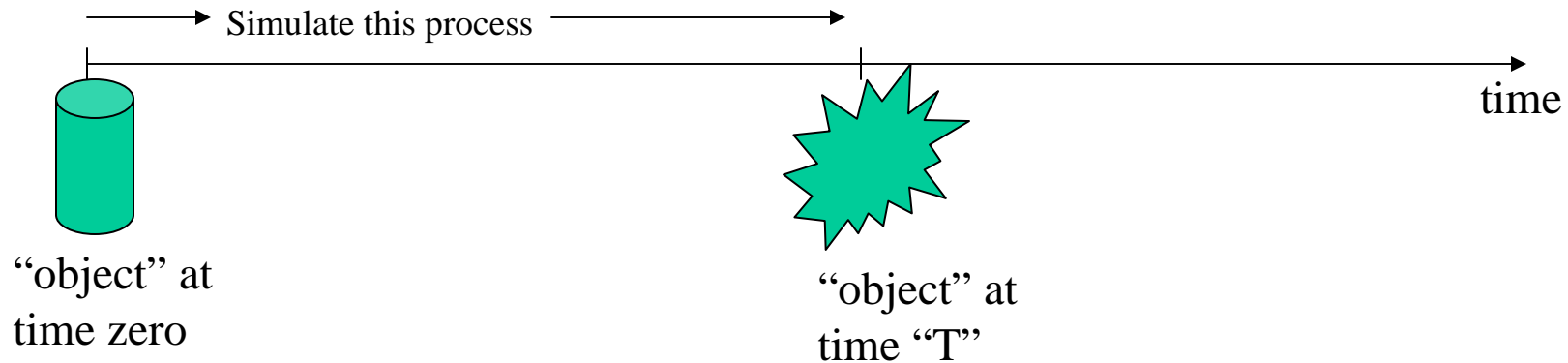
Weapons Simulation: The Numerical Challenge



Numerical considerations:

1. Complex time dependent, non-linear, coupled differential equations
2. 3D
3. Enormous matrices to be “solved”
4. Quantify uncertainty without experiments

Weapons Simulation: The Computational Challenge



1. Huge codes:
 - a. 0.5-1.0 million lines of code, massively parallel
 - b. Complex SQA problems
 - c. Code components in multiple languages
2. Really BIG massively parallel computers (tens/hundreds of tera-ops)
3. Massive I/O (data transfer) challenges
4. Visualization challenges
5. Language issues
6. Cosmic radiation (keep the nodes running)

Hydrodynamics: the momentum equation

$$(\mathbf{F} = \mathbf{MA})$$

$$\frac{d\vec{v}}{dt} = -\frac{1}{\rho} \vec{\nabla} p$$

Differential eqn => difference equation:

How many spatial cells to resolve? $1000/D? \Rightarrow 1000^{**3}$
How many time steps to resolve? 10,000? } 10^{**13} dp*

That's a pretty BIG problem! A minimum 3D problem needs $\sim 10^{**10}$ dp

The historic solution....reduce the dp's to fit the computer and your lifestyle.

* dp = discretization points

Neutron transport: the Boltzmann equation

$$\frac{1}{v(E)} \frac{\partial \psi(\vec{r}, E, \underline{\Omega}, t)}{\partial t} + \underline{\Omega} \cdot \vec{\nabla} \psi(\vec{r}, E, \underline{\Omega}, t) + \sigma_t(\vec{r}, E, t) \psi(\vec{r}, E, \underline{\Omega}, t) =$$

$$\int \int_{4\pi} \sigma_s(\vec{r}, E' \rightarrow E, \underline{\Omega}' \rightarrow \underline{\Omega}, t) \psi(\vec{r}, E', \underline{\Omega}', t) dE' d\Omega' +$$

$$\frac{\chi(E)}{4\pi} \int_0^\infty v(\vec{r}, E', t) \sigma_f(\vec{r}, E', t) \phi(\vec{r}, E', t) dE' + S_{ext}(\vec{r}, E, \underline{\Omega}, t)$$

Differential eqn => difference equation:

How many spatial cells to resolve? $1000/D? \Rightarrow 1000^{**3}?$

How many energy groups to resolve? 100?

How many angular groups to resolve? 100?

How many time steps to resolve? 10,000?

}

10^{17} dp**

That's a really BIG problem!

The historic solution...reduce the dp's to fit the computer and your lifestyle.

The Transport Equation

**There is a whole lot more to solving the transport equation than dp's!
(re: Jim Morel's talk)**

- **Books have been written on the subject and many careers have been dedicated to the subtleties of the numerics particularly when the dp's are "few" and the dp spacing is complex.**
- **Matrix solvers, pre-conditioners and acceleration schemes are critical.**
- **Getting the limits correct...through thick and thin.**



Some History....the early years

- **1951 - the ENIAC (Von Neumann, LANL) (10^{**4} dp):**
 - the first design code
 - 50 cells, 100 time steps, clock time 6 months
 - language - HPT (holes in paper tape)
- **The 60's - IBM 709,7090, STRETCH (10^{**6} dp):**
 - 10^{**3} cells, 10^{**3} time steps, clock time 10's of hours
 - assembly language
- **1968 - CDC6600 (10^{**7} dp):**
 - Von Neumann's "code"
 - 600 cells, 10^{**4} time steps, clock time 100's of hours
 - FORTRAN
- **1970 - the CDC STAR (LLNL):**
 - vectorization - but not productive



Some History....the mid years

- The early 70's - The CDC 7600 (**10^{**8} dp**):
 - 10^{**4} cells, 10^{**4} time steps, clock time 10's of hours,
- The mid 70's - CRAY 1 (**10^{**9} dp**):
 - vectorization works!
 - 10^{**5} cells, 10^{**4} time steps, clock time 10's of hours,
- 1981
 - Salishan #1
 - WPC “designs” parallel CRAY architecture for Monte Carlo!
 - Vectorization of Monte Carlo was difficult (but doable) => Bill Martin, Forrest Brown
 - Parallel architecture seems a natural for Monte Carlo
- The late 80's - Cray XMP, YMP (**10^{**10} dp**):
 - Parallel architecture
 - 10^{**5} cells, 10^{**4} time steps, 10's of hours
 - but codes still mostly serial

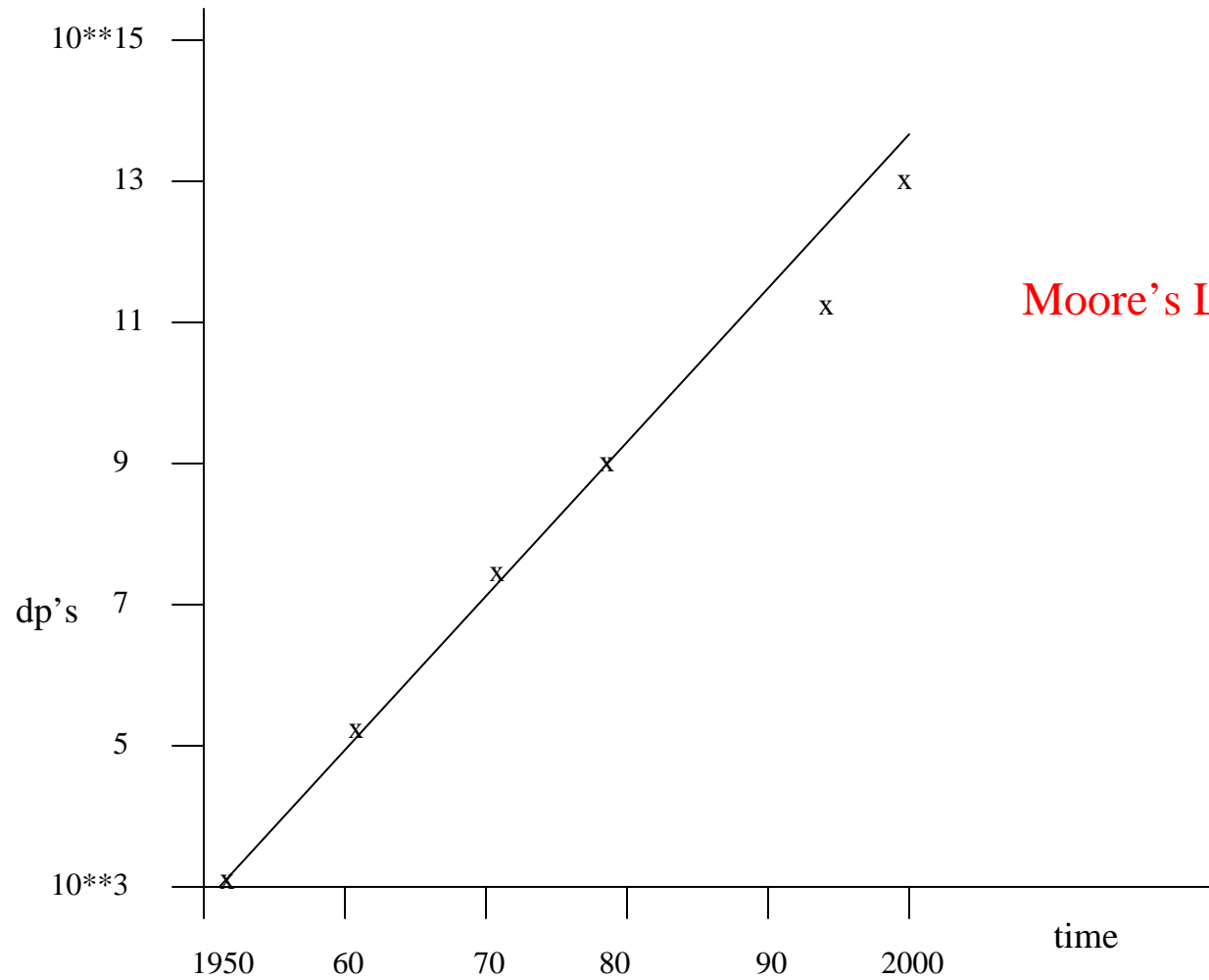


Some History....the recent years

- **The early 90's - Connection Machine (LANL), the Meiko (LLNL) and others:**
 - Nuclear testing ends
 - No vectorization
 - some venturing into parallel processing
 - Not heavily used by design codes
 - Code porting issues
- **The mid 90's - DEC Alpha Clusters (LLNL), UNIX workstations:**
 - Vectorization goes away
 - codes ported to UNIX workstations (the end of LRLTRAN, LTSS)
 - C, C++ for a few but FORTRAN for most
 - Some parallelism, but minimal
- **The late 90's - ASCi Blue (SGI, IBM) (10^{11} dp):**
 - Lots of \$\$\$ being spent
 - Emphasis on 3D
 - 10^6 cells, 10^4 time steps, clock time 100's hours
 - Full parallel implementation - finally!
- **The early 2000's - ASCi HP-Compaq Q (10^{13} dp):**
 - 10^8 cells, 10^4 time steps, clock time 100's hours



Dp's vs time



Moore's Law ?!

The Typical ASCI Code

- **0.5 - 1.0 Million lines**
- **Spatial discretization may be extremely complex**
- **Fortran 77 - 95, C, C++**
- **Planned 25-40 year life cycle**
 - **Expected moderate rewrites every ~15 years**
- **MPI, maybe some threading**
- **Totalview debugger (and “print”)**
- **Of order 10 terabytes of output data per BIG run**



User ASCI experiences

- Turn around for many problems - less than a day.
- Turn around for “hero” problems – week(s) to month(s).
- Typically months of debugging and trial runs prior to the “successful” hero run.
- The goal: Provide answers while we still remember what we’re looking for.

Historically speaking: we need to turn problems around in less than a week (~ 50 hrs of machine time) if not a day!

A nuclear design study could easily require 10^{3} runs!**



What do we need (architecture)? (1)

- **Fast CPUs**
 - not all problems are communications bound
- **Lots of memory**
 - would like a homogenous resource
 - some problems using 2GB/PE
- **High speed interconnect network**
 - “maintain” ratio (processor speed/network speed) 10-20 Flop/Byte
- **Truly global parallel file system**
 - data on PFS to be visible from any processing element in the computing center



What do we need (architecture)? (2)

- **High bandwidth secondary storage (including the supporting network)**
 - How do we efficiently save data without slowing down on-going processing?
 - generating up to 200 GB per restart dump (every 30-60 minutes).
- **High Capacity Visualization**
 - tied to data movement issues, real-time processing, and data reduction
 - generating 140 GB of data per time slice
- **Robustness**
 - (see next page)

Robustness issues

- **Example: on Q series ASCi machines, Mean Time To Interrupt ~ 6 hours.**
 - MTTI might become more of an issue as we scale to PetaFlop processing using a cluster approach with many more parts that can randomly fail.
- **Scheduling of resources - a painful issue.**
- **User productivity**
 - <100 PEs: very high (+90%)
 - >1000 PEs: not so high (~50%)
- **Frequent restart dumps and automated submission/monitoring scripts ease the pain.**



What have we learned?

- **We will use all the capacity that is provided.**
- **Some of our applications scale well to 1000's of PEs.**
 - Two of our main ASC codes scale optimally to at least 1800 PEs.
- **We are a flexible and inventive bunch.**
 - We will overcome or at least minimize shortcomings in any architecture.
 - We waste a lot of time overcoming.
- **Keep it standard! We can't spend years porting our codes.**
 - UNIX
 - MPI
 - ...



In reflection....

- **Fast CPU's - YES**
- in conjunction with a fast network interconnect
- **Lots of memory - YES**
- **Vectorization - YES...but keep it “simple” to implement**
- **Parallelization - YES...but it's getting very complicated!**
- **Languages - most of us (code physicists) still prefer FORTRAN and C (I think procedures) but I do like the heirarchal OO design.**
- **Debuggers - are critical; appreciate any help we can get to improve them.**
- **FLOPS - what comes after “peta”?**



The End



Salishan Conference on High Speed Computing - 2004
04/20/04 - 22

