SAND2011-1386C

Preparing Multi-physics, Multi-scale Codes for Hybrid HPC

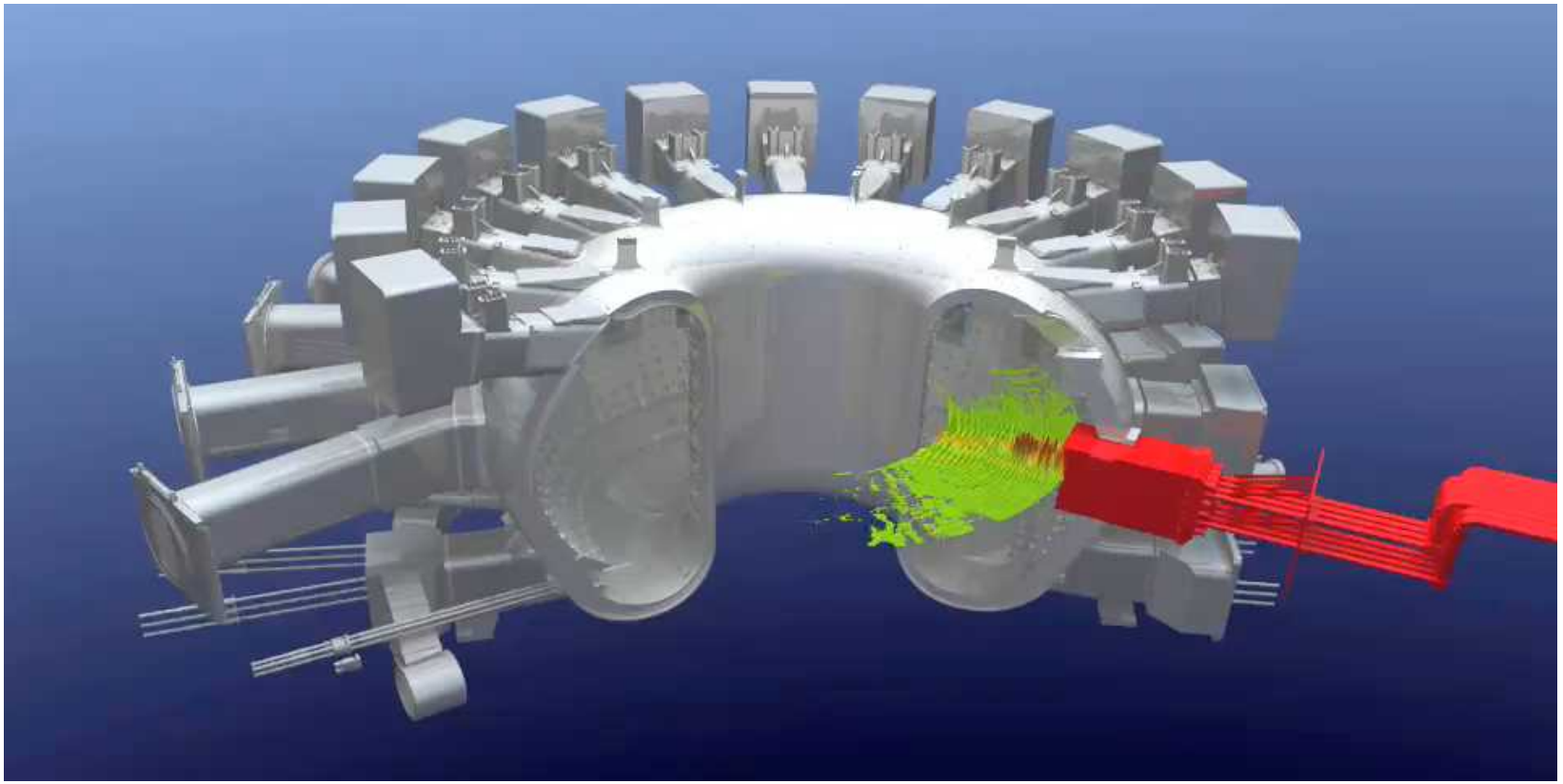
March 3, 2011

Richard Barrett, Rich Drake, and Allen Robinson
Center for Computing Research (1400)

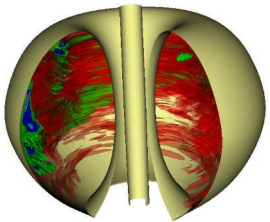
Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

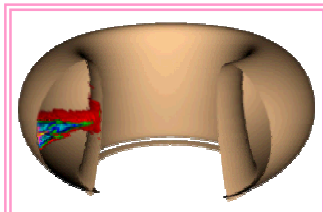
Programming model, mechanisms, etc

- How programmer views data and the computations that operate on it.
- Mechanism: MPI, OpenMP, cuda, opencl, etc
- Critical link: how machine views data and the computations that operate on it.
- Over-arching goal: science and engineering



AORSA simulation;
movie by Sean Ahern@ORNL





Sandia National Laboratories

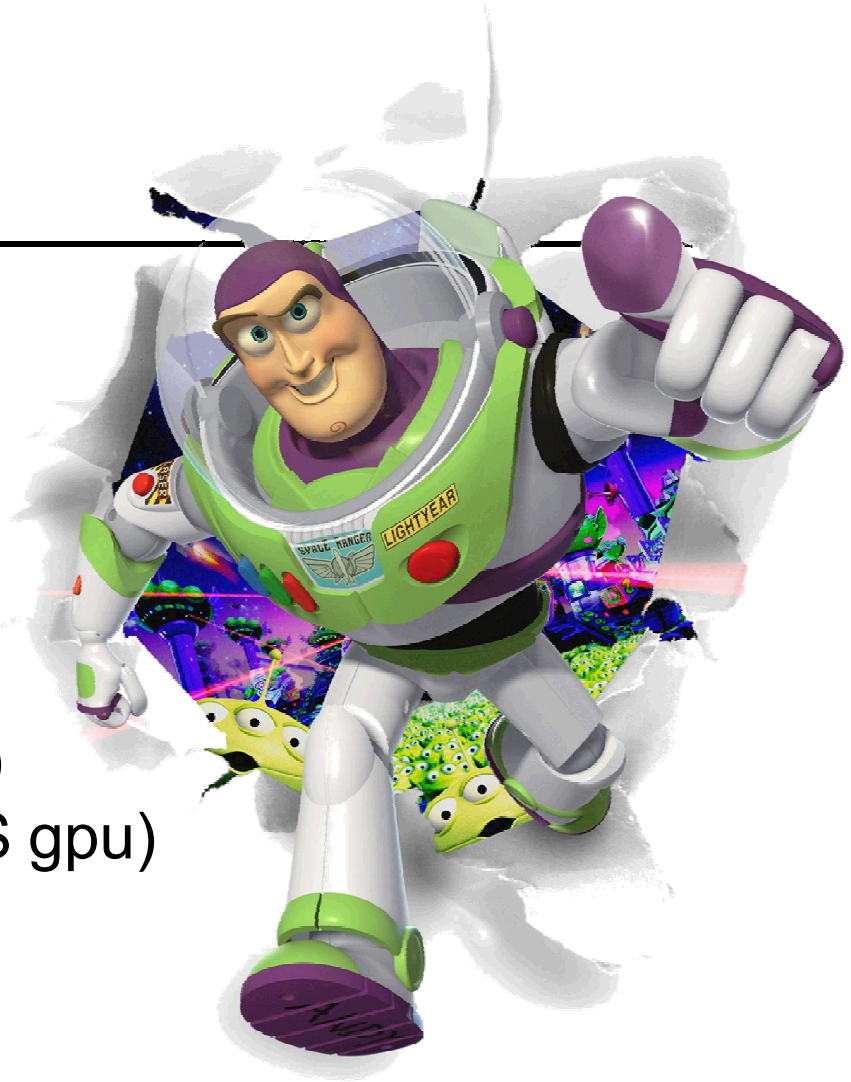
```
C     APPROXIMATE VALUES FOR SOME IMPORTANT MACHINES ARE:
C
C         IBM/195    CDC/7600  UNIVAC/1108   VAX 11/780 (UNIX)
C          (D.P.)  (S.P.,RNDG)    (D.P.)     (S.P.)     (D.P.)
C
C NSIG      16         14          18           8         17
C ENTEN   1.0D75     1.0E322     1.0D307     1.0E38    1.0D38
C ENSIG   1.0D16     1.0E14      1.0D18      1.0E8     1.0D17
C RTNSIG  1.0D-4     1.0E-4      1.0D-5      1.0E-2    1.0D-4
C ENMTEN  2.2D-78    1.0E-290    1.2D-308    1.2E-37   1.2D-37
C XLARGE  1.0D4      1.0E4       1.0D4       1.0E4     1.0D4
C EXPARG  174.0D0    740.0E0     709.0D0     88.0E0    88.0D0

c timing              on ncar"s control data 7600, besic takes about
c                   .32+.008*n milliseconds when z=(1.0,1.0).
c
c portability          ansi 1966 standard
```

Sandia National Laboratories

Target architectures

- Small clusters: linux, SunOS, IRIX, AIX
- MPP: Red Storm, Red Sky
- New ASC capability: Cielo

…and beyond



Peta-scale is 150k Opterons, and clocks are not increasing, so exa-scale is 150M Opterons? BlueGene even more?

“Accelerator”-based arch

- Cell + Opterons (Roadrunner)
- gpu + x86 (nVidia: 1 TFLOPS gpu)
- LCF3@ORNL: 20PF, mc+gpu

Intel many-core, eg Knights Ferry

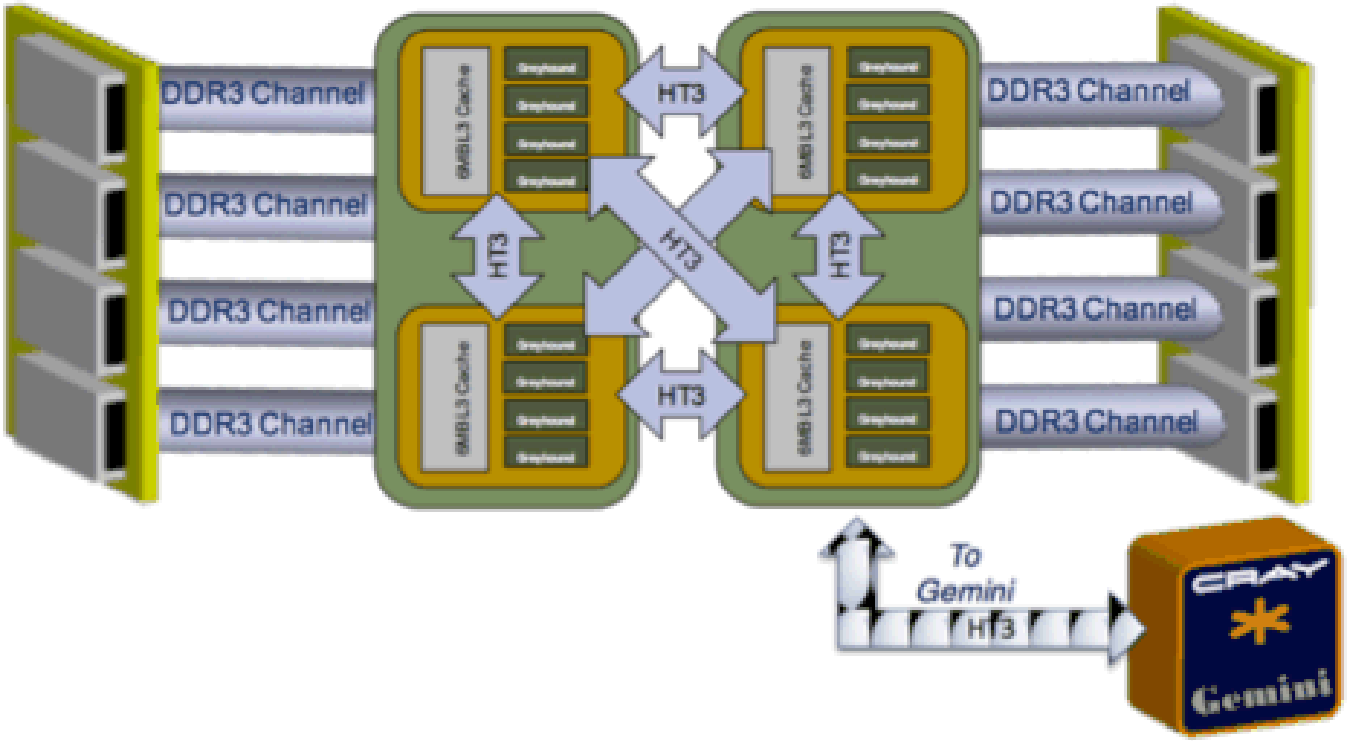
So how do we program these?!?!

Goal :
At most, one and a half code re-writes

1: Revolutionary: programming model

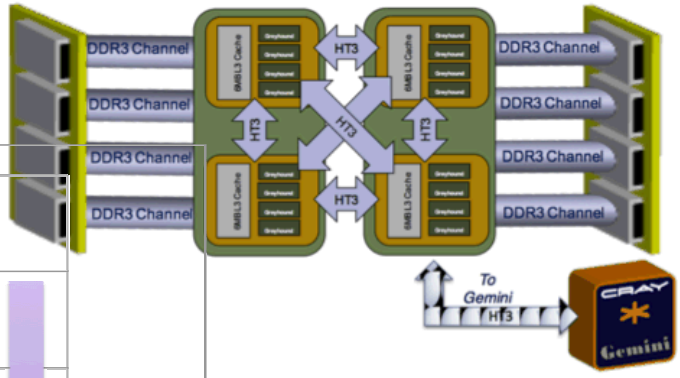
½ : Evolutionary: programming mechanism

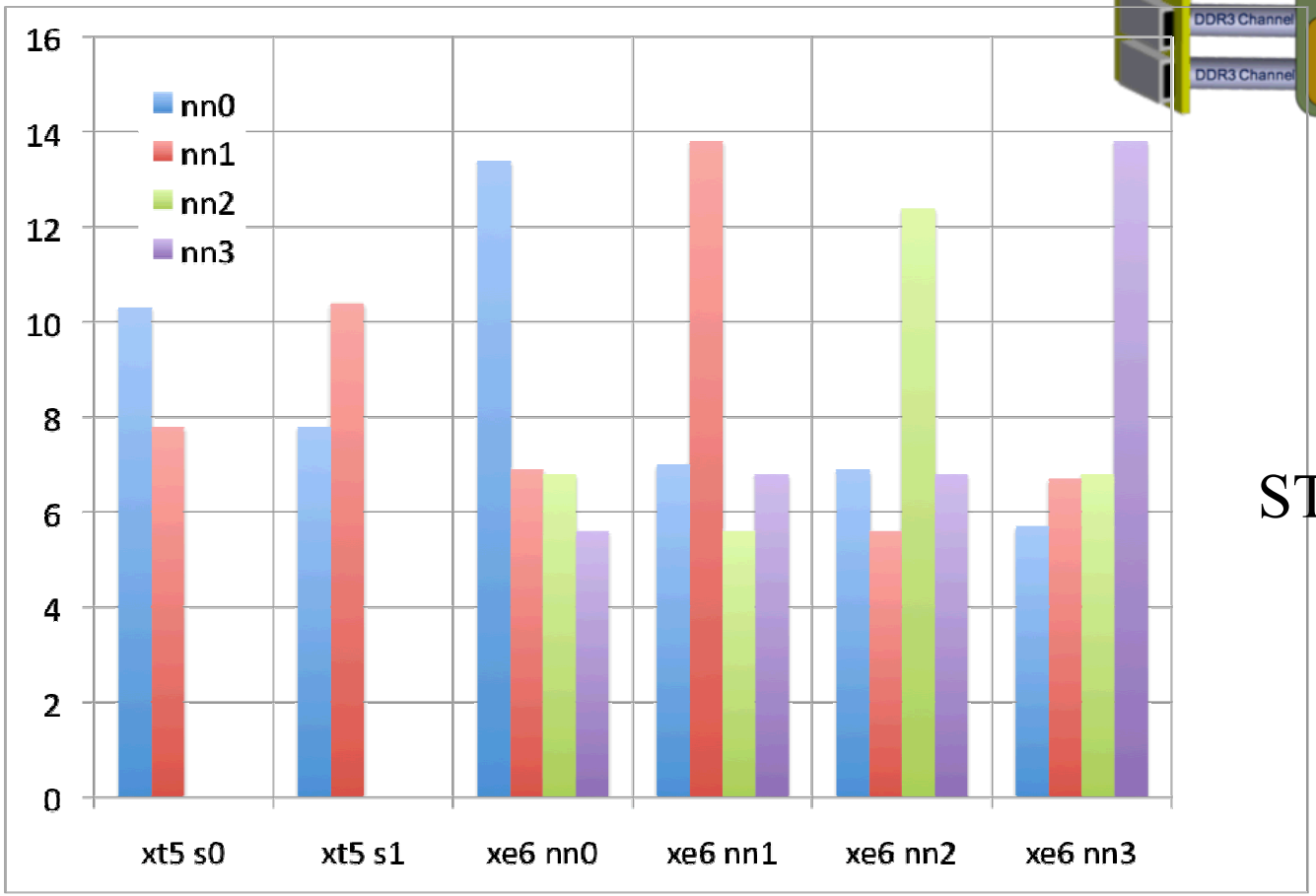
Cielo Cray XE6



Sandia National Laboratories

Cielo Cray XE6

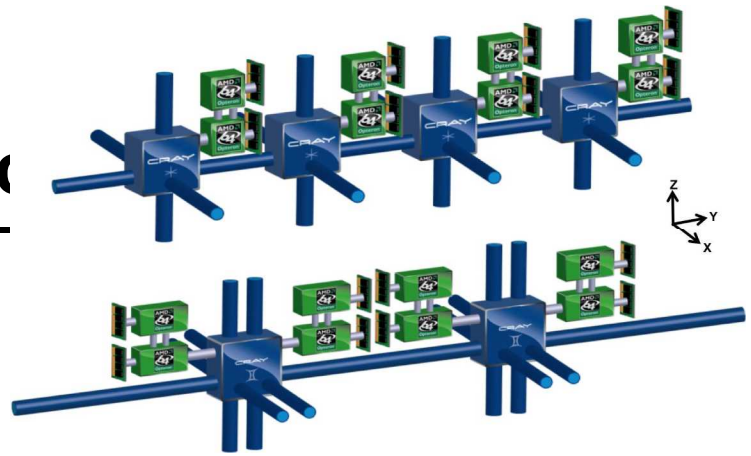


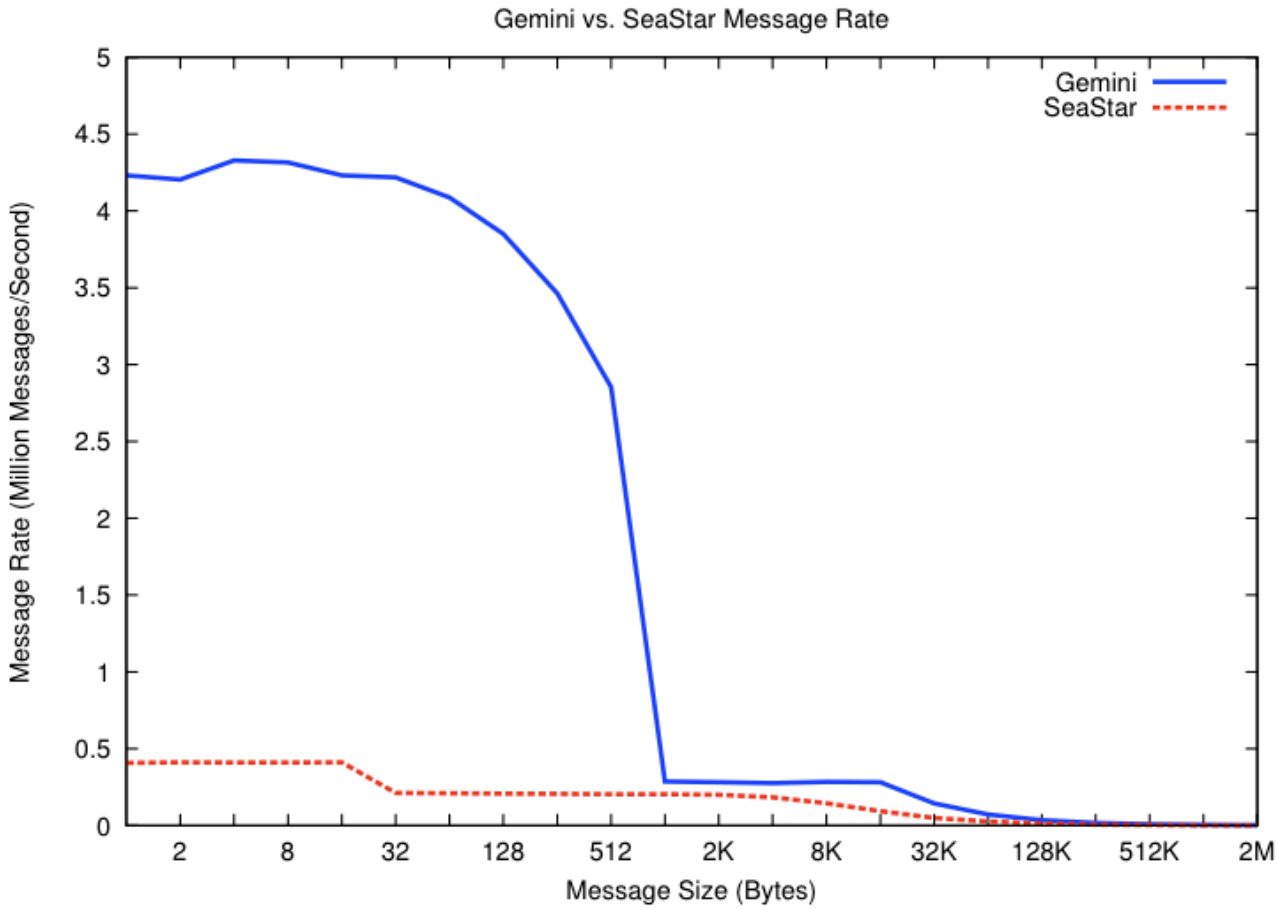


STREAM TRIAD
GB/sec

Sandia National Laboratories

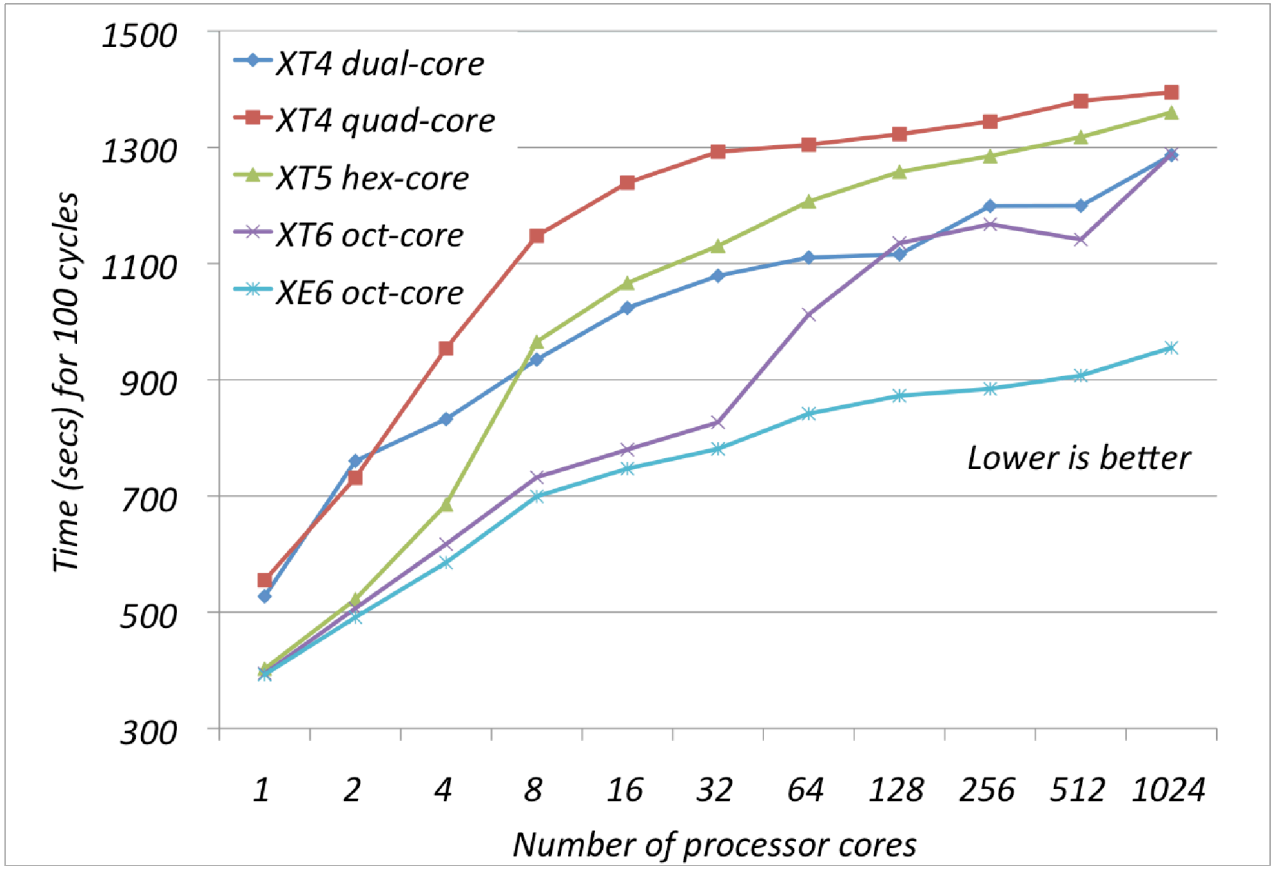
Cielo Gemini Interc





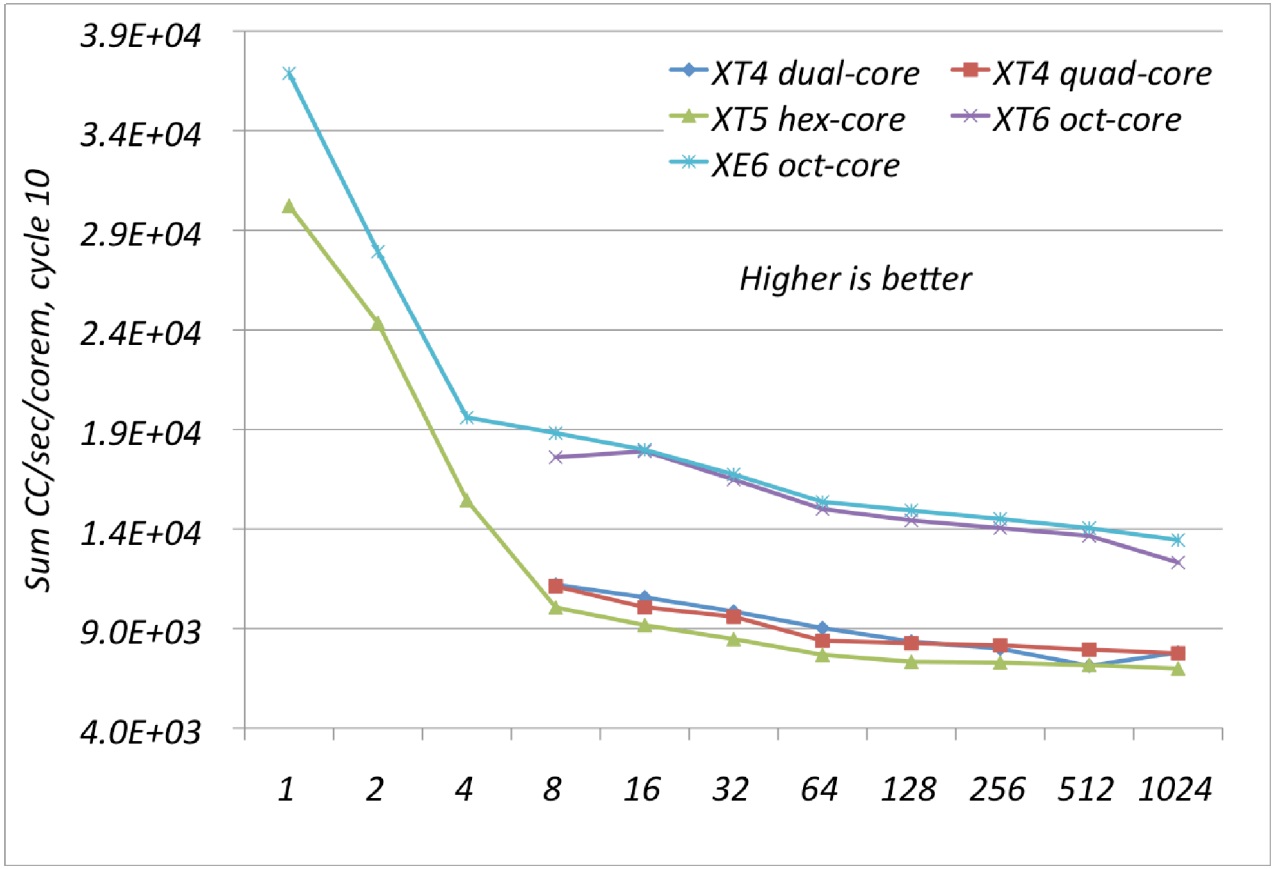
Sandia National Laboratories

CTH



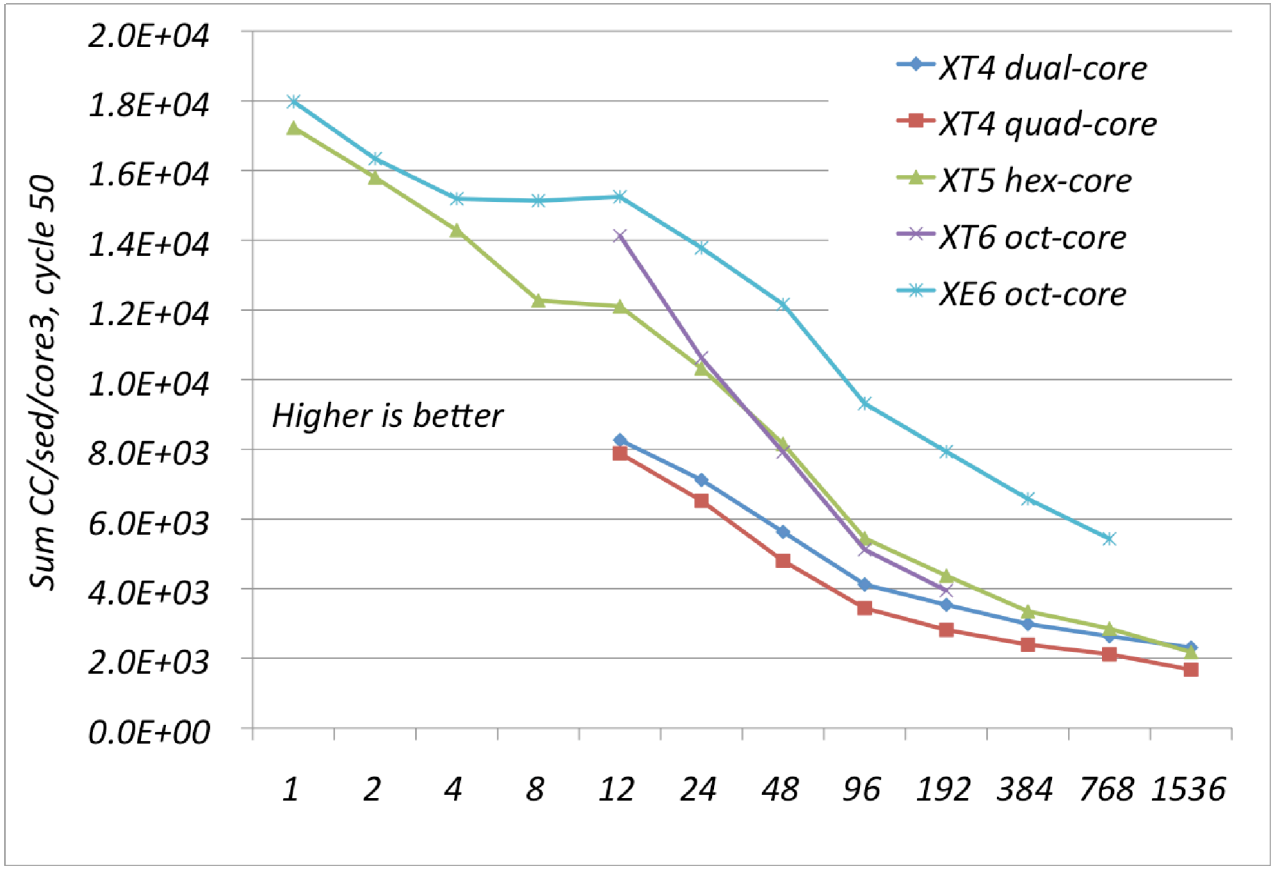
Sandia National Laboratories

SAGE



Sandia National Laboratories

xNobel


2.0E+04
1.8E+04
1.6E+04
1.4E+04
1.2E+04
1.0E+04
8.0E+03
6.0E+03
4.0E+03
2.0E+03
0.0E+00
Sum CC/sed/core3, cycle 50
Higher is better
XT4 dual-core
XT4 quad-core
XT5 hex-core
XT6 oct-core
XE6 oct-core
1
2
4
8
12
24
48
96
192
384
768
1536

What's up with all those languages?

```
! Fortran-MPI
call get_ghosts ( )
C = A + B
```

```
/* UPC */
upc_forall( i=0; i<n; i++; &a[i] )
   c[i] = a[i] + b[i];
```

```
! Fortran-shmem
SHEM_PUT GET_GHOSTS (…)
C = A + B
```

```
// Chapel
forall i in D do
  c(i) = a(i)+ b(i)
```

```
! Co-array Fortran
do i = 1,n
   c(i) = a(i)[j(i)]* b(i)[j[i]]
enddo
```

```
! PGI-GPU
!$acc region
C = A + B
!$acc end region
```

```
// CUDA is C plus a little stuff
int i = blockIdx.x * blockDim.x + threadIdx.x;
if (i < n)
   y[i] = alpha * x[i] + y[i];
```

```
% Star-P
C = A + B;
```

Sandia National Laboratories

New Language Acceptance: History of Fortran

1953	Proposal to IBM
1954	Draft spec
1956	Manual
1957	Compiler
1960	Compilers for IBM 709, 650, 1620, 7090
1962	> 40 compilers

HPCS langs are (sorta) here

IBM 7090 at NASA Ames

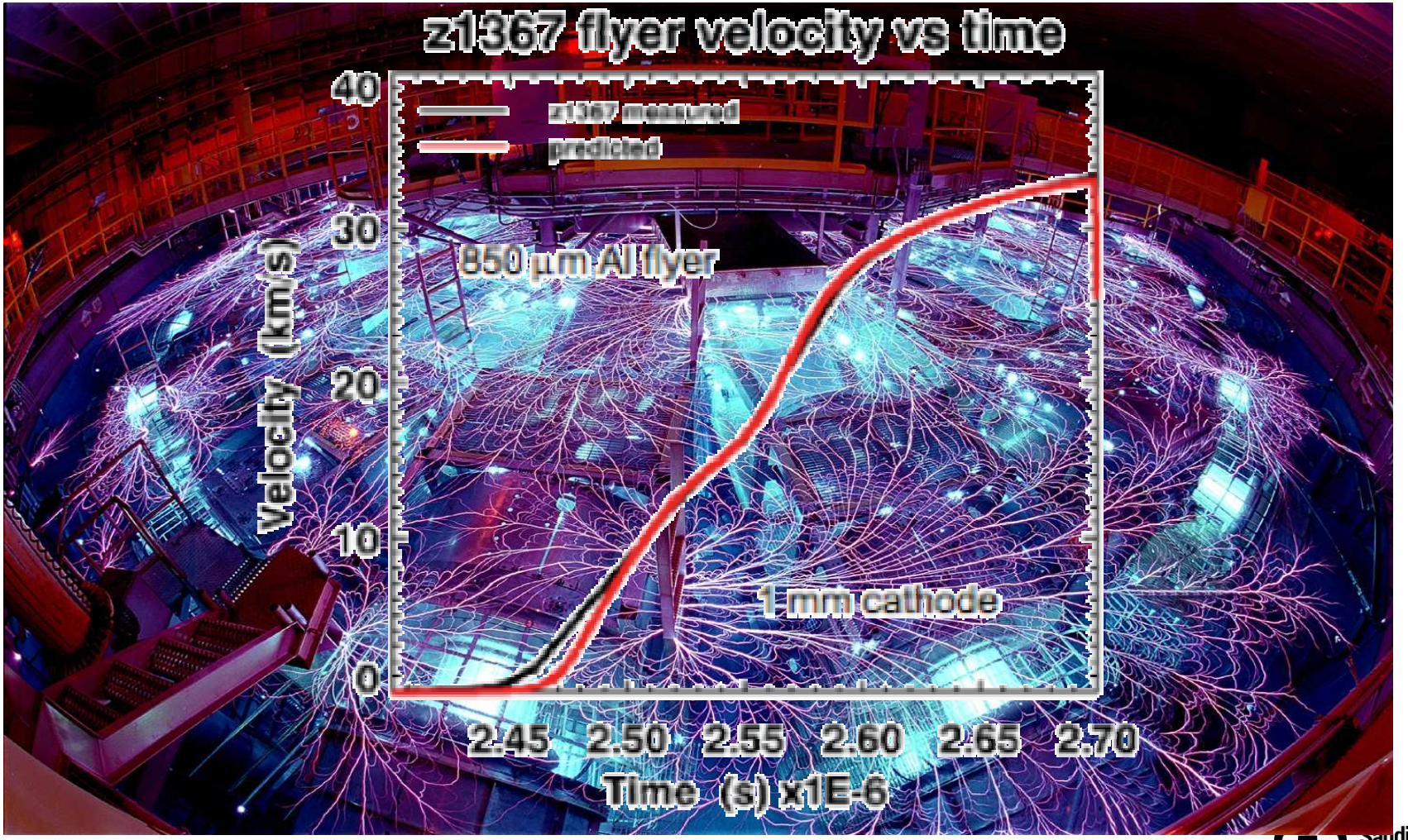
ALEGRA

- Simulate large deformations and strong shock physics including solid dynamics in an Arbitrary Lagrangian-Eulerian methodology

- Also magnetics, MHD, electromechanics and a wide range of phenomena for high-energy physics applications.

ALE

Pulsed power: Z-machine



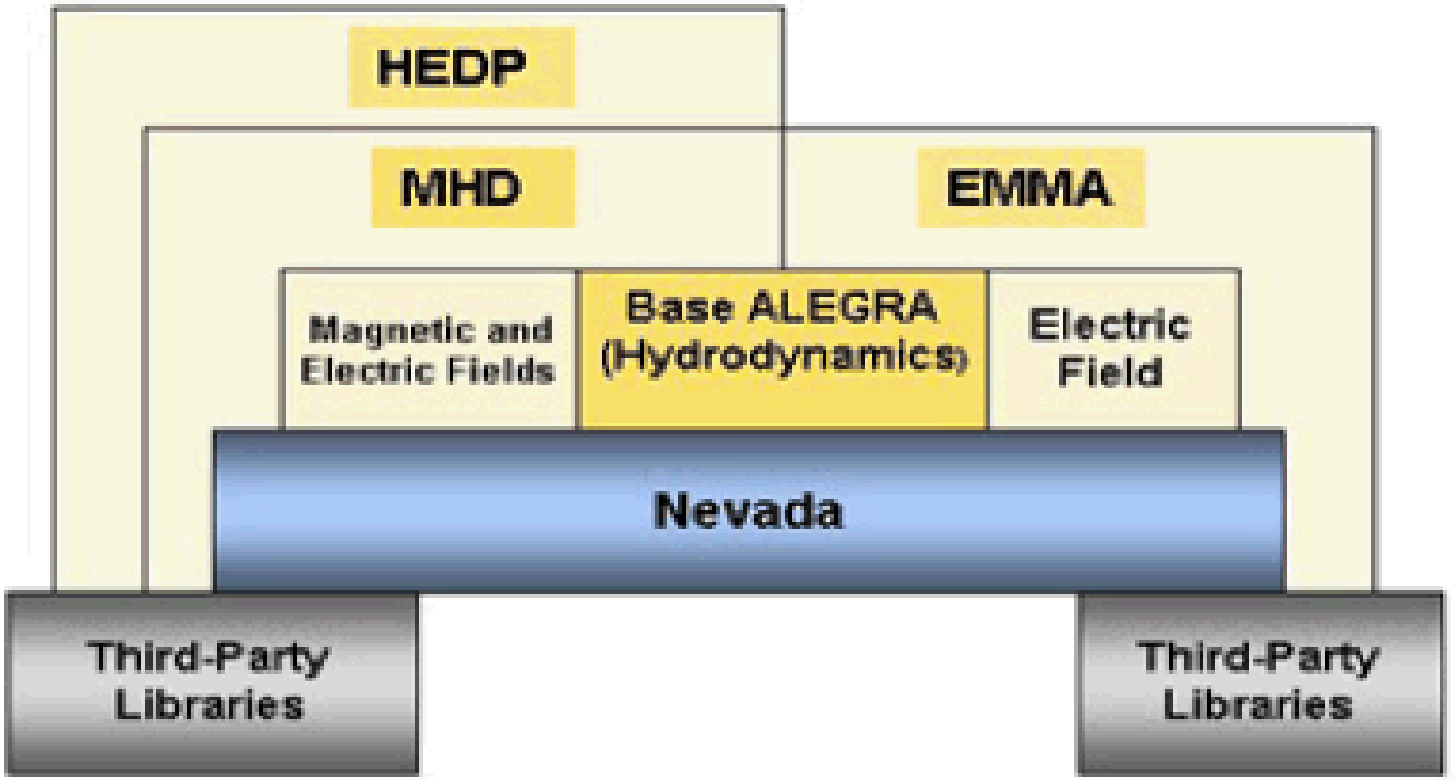
Sandia National Laboratories

ALEGRA code base*
est. 1990

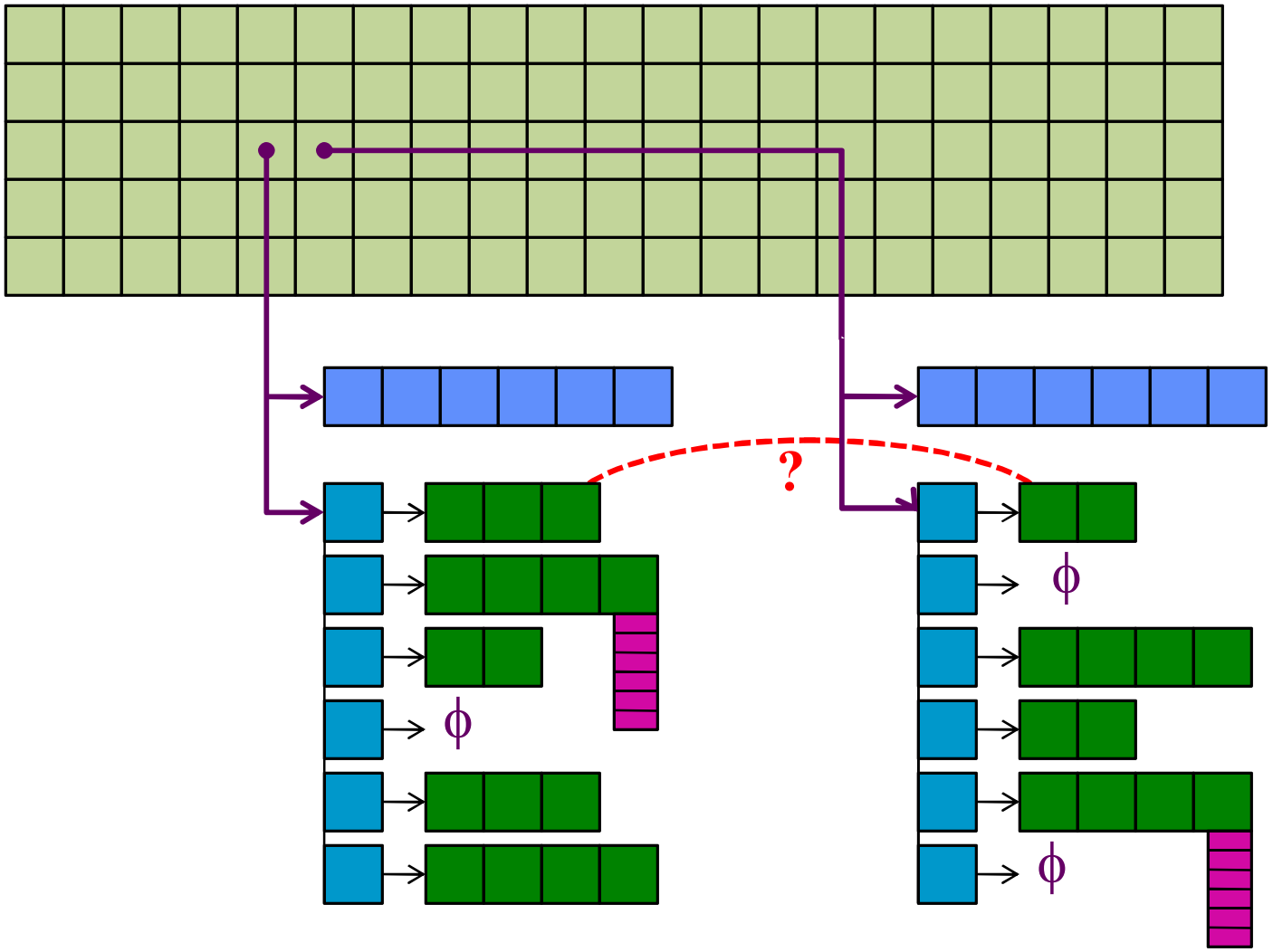
```
                    C/C++ SOURCE LINES OF CODE COUNTING PROGRAM
          ( c ) Copyright 1998 - 2000 University of Southern California, CodeCount (TM)

   University of Southern California retains ownership of this copy of software.  It is licensed to you.
Use, duplication, or sale of this product, except as described in the CodeCount License Agreement, is
strictly prohibited.  This License and your right to use the software automatically terminate if
you fail to comply with any provisions of the License Agreement.  Violators may be prosecuted.
This product is licensed to : USC CSE and COCOMO II Affiliates


The Totals
   Total    Blank |      Comments      |  Compiler  Data    Exec.   |  Number   |          File  SLOC
   Lines    Lines |   Whole  Embedded  |  Direct.   Decl.   Instr.  | of Files  |   SLOC   Type  Definition
-------------------------------------------------------------------------------------------------------
--------------------
  388275    62268 |   72506      8267  |    14688   64562   174252  |   1241    |  253502  CODE  Physical
  388275    62268 |   72506      8267  |    14622   32912   116441  |   1241    |  163975  CODE  Logical
    5388      778 |       0         0  |        0    4610        0  |     68    |    4610  DATA  Physical

Number of files successfully accessed.....................  1309 out of 1353

Ratio of Physical to Logical SLOC.........................     1.55

Number of files with :
        Executable Instructions        >     100          =     289
        Data Declarations              >     100          =      48
        Percentage of Comments to SLOC <      60.0 %      =     697      Ave. Percentage of Comments to Logical
 SLOC =  49.3

REVISION AG4  SOURCE PROGRAM -> C_LINES          This output produced on Wed Feb 23 10:20:26 2011
```

** Excluding some Fortran (58k@121f), python, xml, etc, some uncounted files, and the Nevada framework.*

Sandia National Laboratories


HEDP
MHD
EMMA
Magnetic and Electric Fields
Base ALEGRA (Hydrodynamics)
Electric Field
Nevada
Third-Party Libraries
Third-Party Libraries

ALEGRA data structure



Sandia National Laboratories

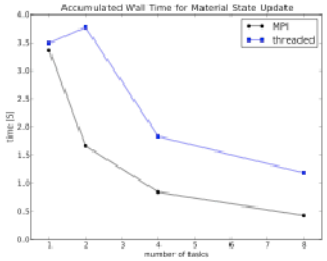
```
…bunch of code (including
 accesses of element data)…
loop on all elements:
  …bunch of code…
  call material update
```



```
…bunch of code (including
 accesses of element data)…
#pragma omp parallel for
for (i=0;i<numthreads;++i)
  loop on all elements in list[i]:
    …bunch of code…
    call material update
```

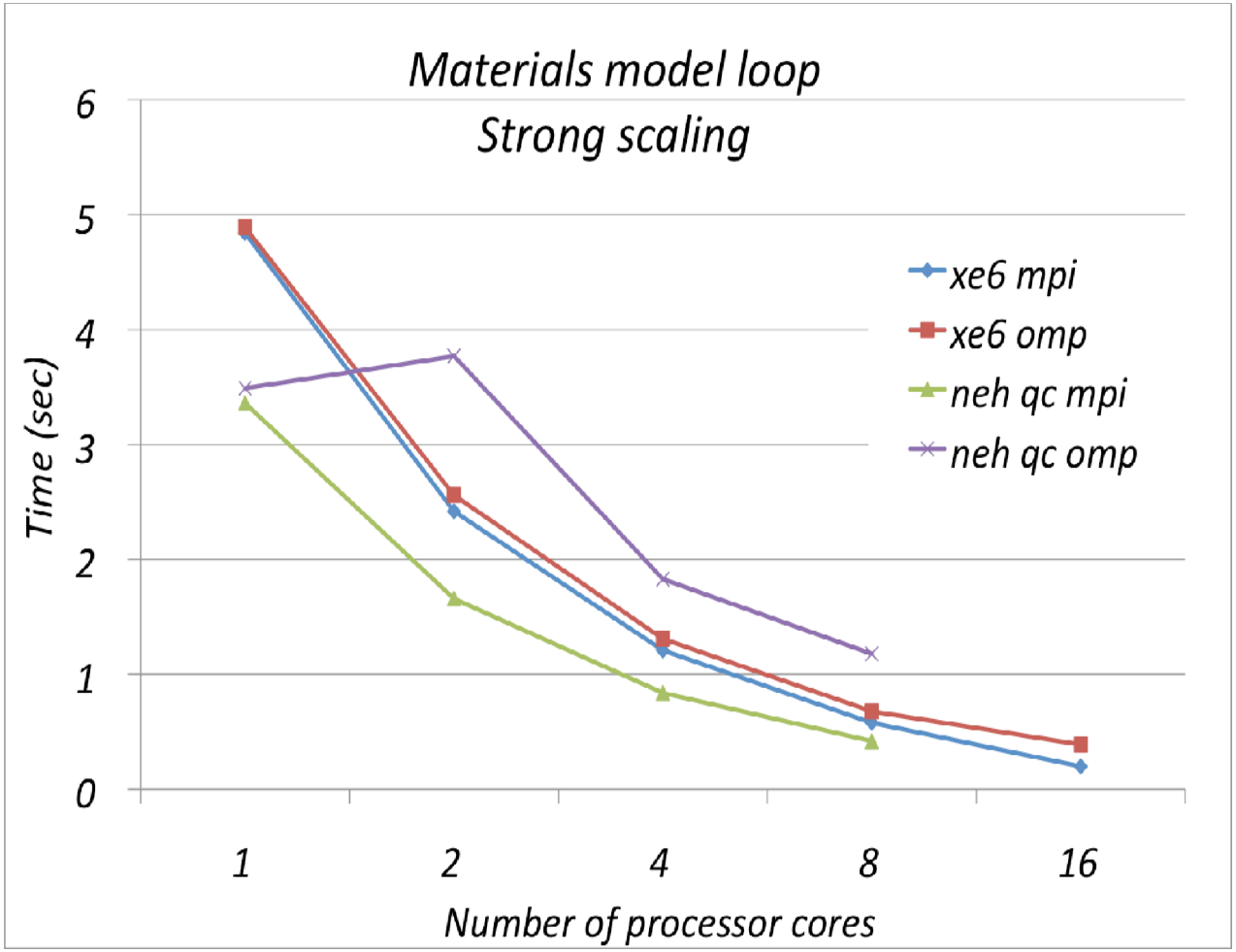
I modified the alegra/src/uns_dynamics.C file in the function UnsDynamics::Material_State_Update() to break up the current element list iteration into chunks for each thread. Then used an OpenMP pragma for the element loop. I also used ifdef's so the code will compile without OpenMP. I compiled the code using GCC 4.1 and ran the smlag problem and collected the following timings on my 8 processor RHEL5 CEE LAN workstation. (These are the same as a redsky node.) Strong scaling (total problem size kept the same, 40000 elements.) Raw data:

nprocs	MPI	threaded
1	3.36	3.49
2	1.66	3.77
4	0.84	1.83
8	0.42	1.18



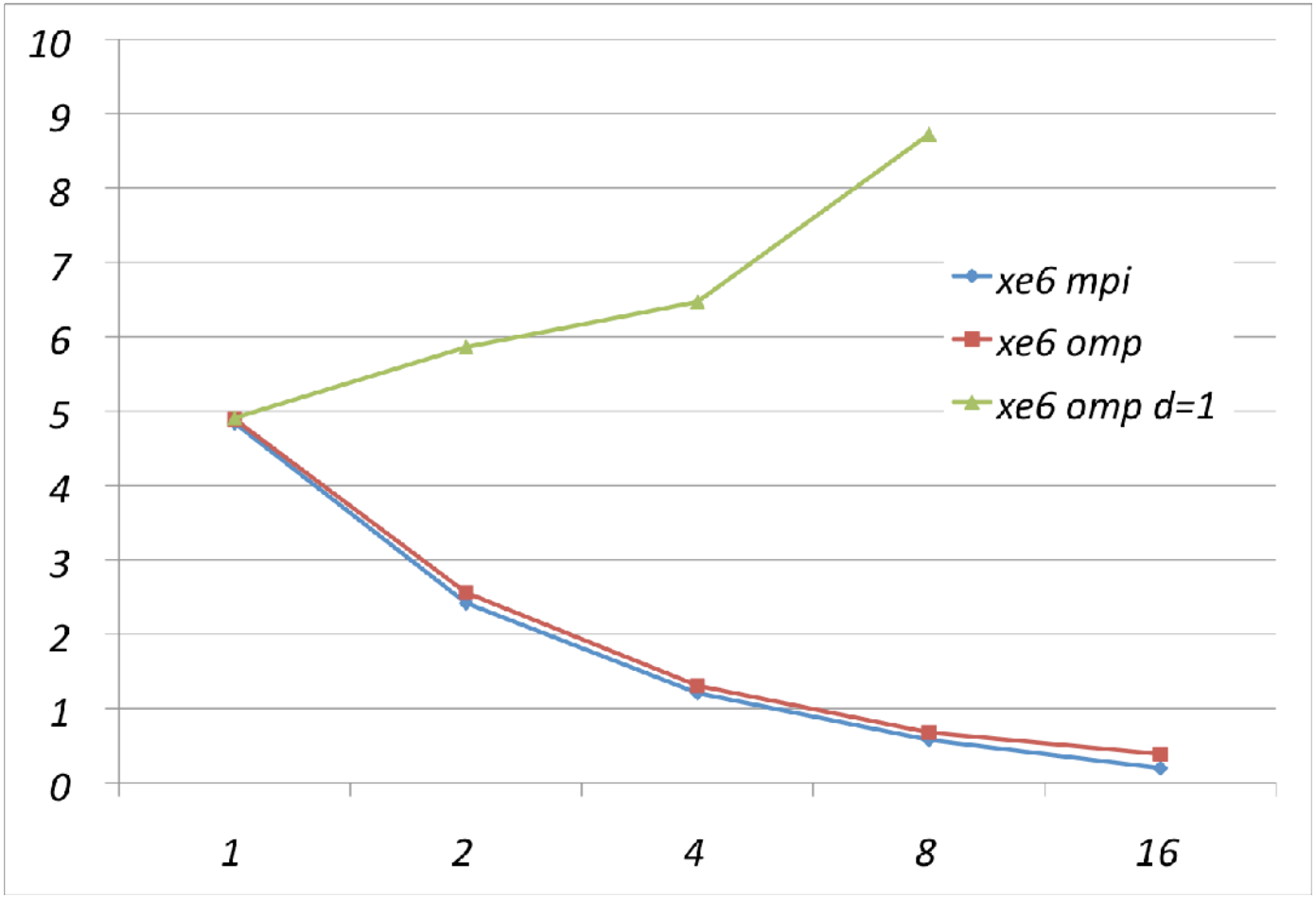
Sandia National Laboratories

ALEGRA threading experiment (*Preliminary work*)



Sandia National Laboratories

Don’t forget to read man pages!



Sandia National Laboratories

Acknowledgements

- Sandia CSRF
- ASC CSSE

Sandia National Laboratories

Thanks

Sandia National Laboratories