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

July 27, 2011

Richard Barrett
Center for Computing Research (1400)

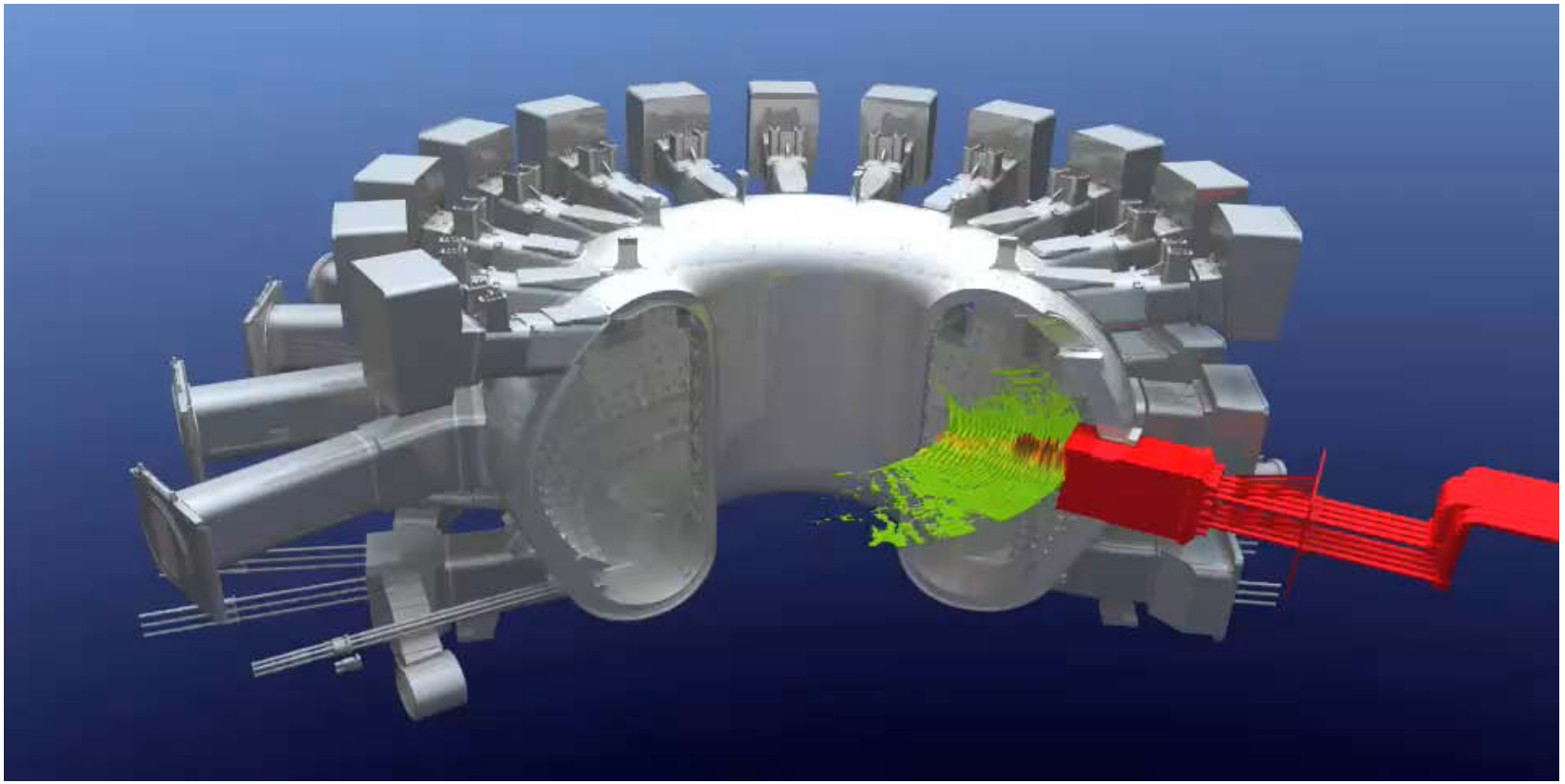
SAND Number 2011-4805 P

OASCR Programming Challenges Workshop

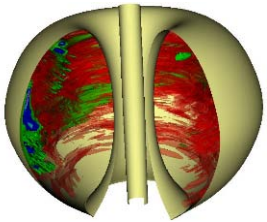
Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

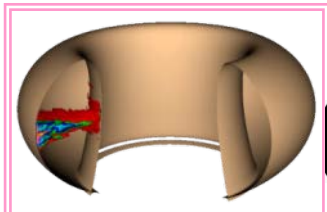
Programming model, mechanisms, etc

- How programmer views data and the computations that operate on it.
- Mechanism: MPI, OpenMP, cuda, opencl, etc
- Critical link: how codesign layers view 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 arc

and beyond!

- Small clusters: linux
- MPP: Red Storm,
- New ASC capabil

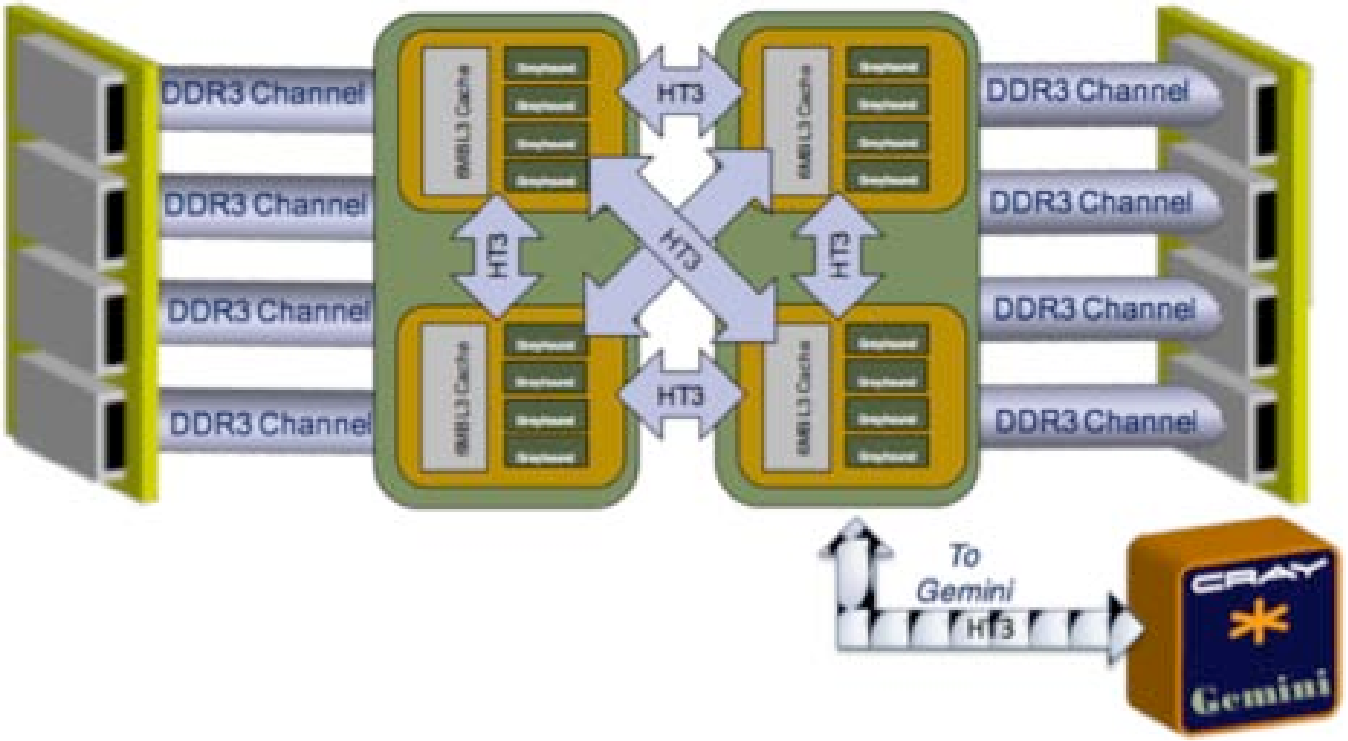
Goal :

At most, one and a half code re-writes

1: Revolutionary: programming model

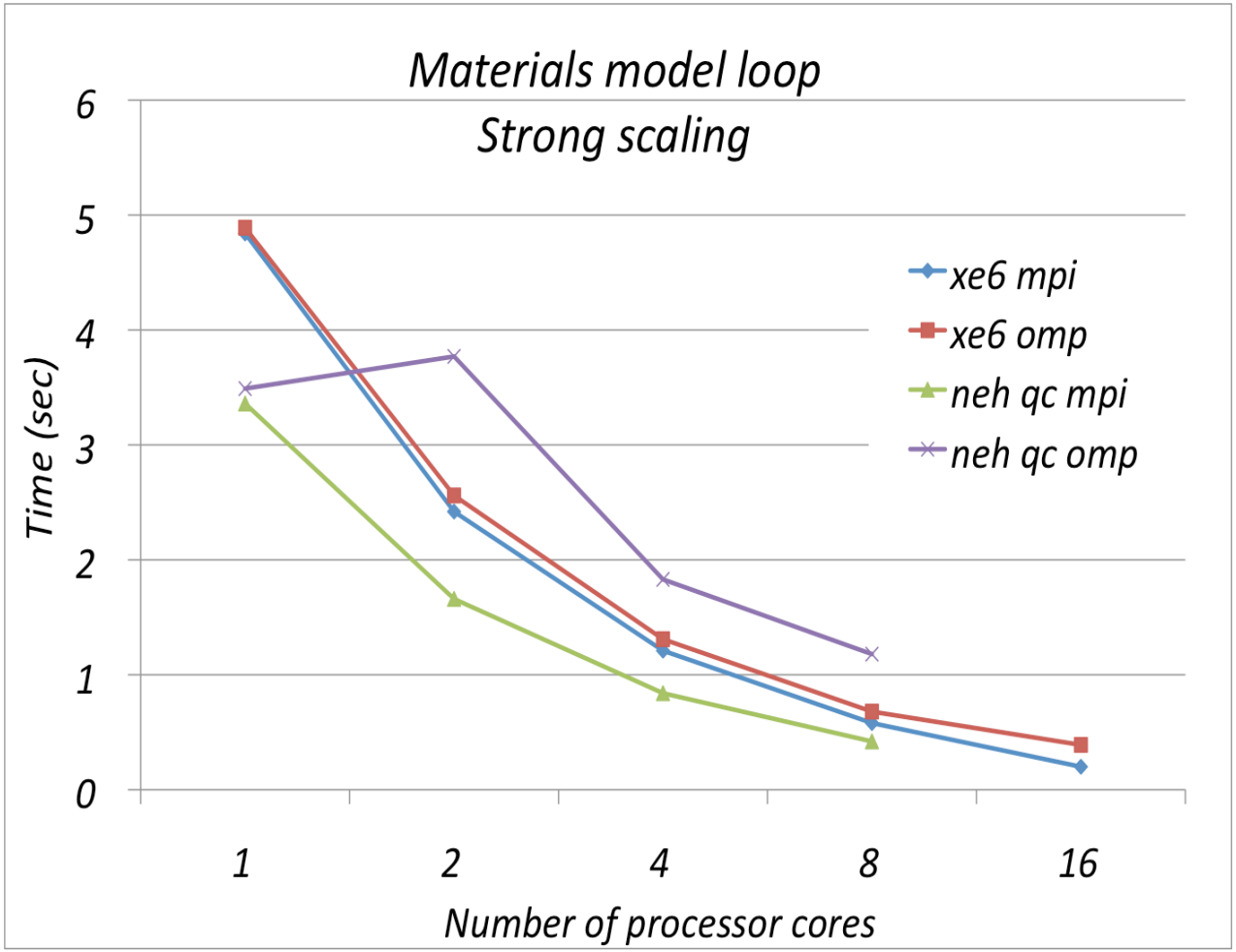
½ : Evolutionary: programming mechanism

Cielo Cray XE6



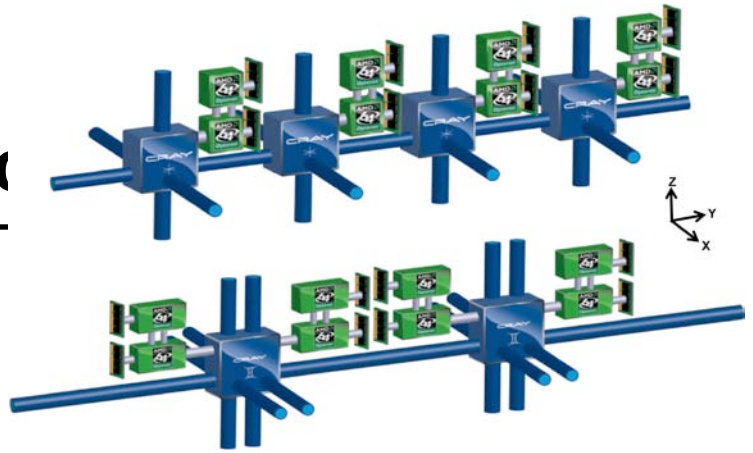
Sandia National Laboratories

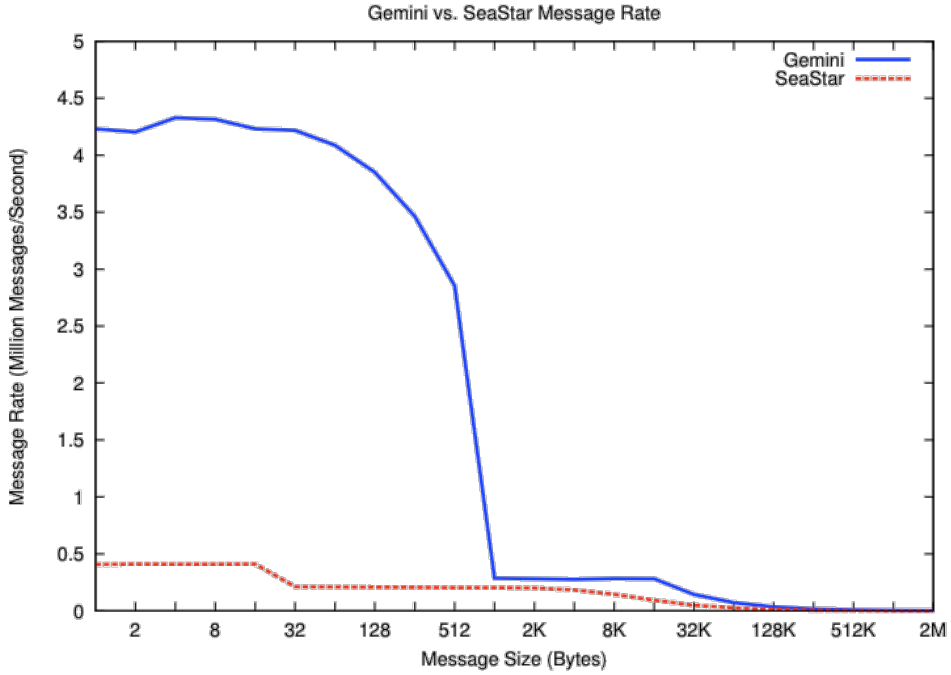
ALEGRA threading experiment (*Preliminary work)*

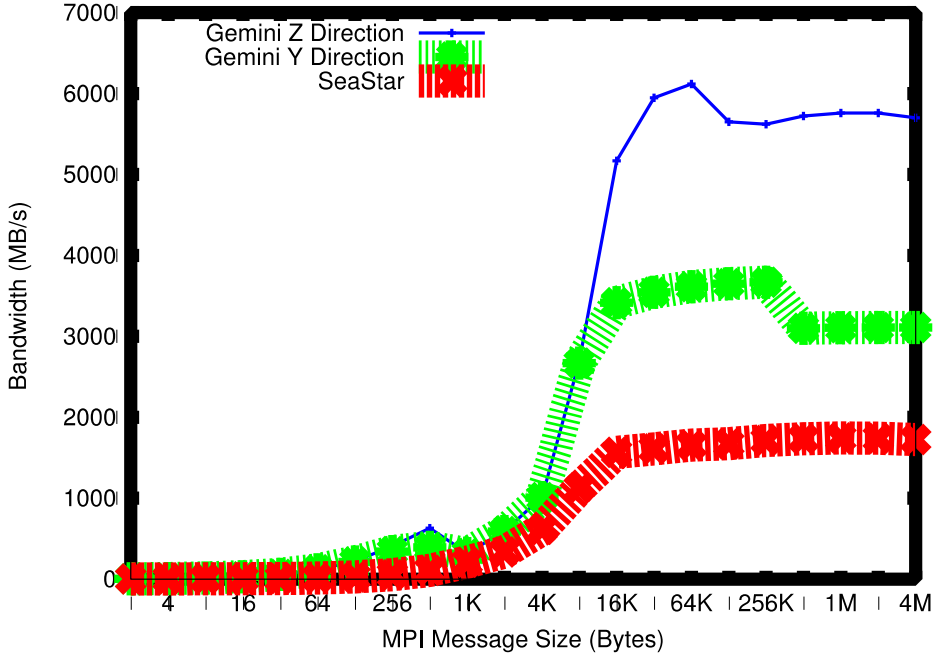


Sandia National Laboratories

Cielo Gemini Interc



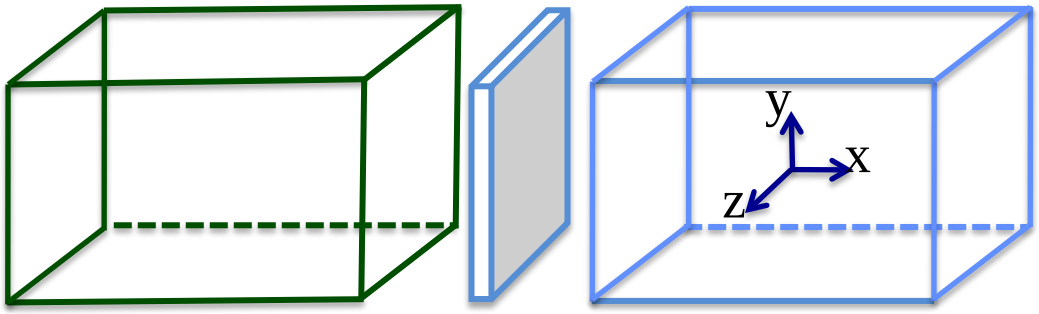




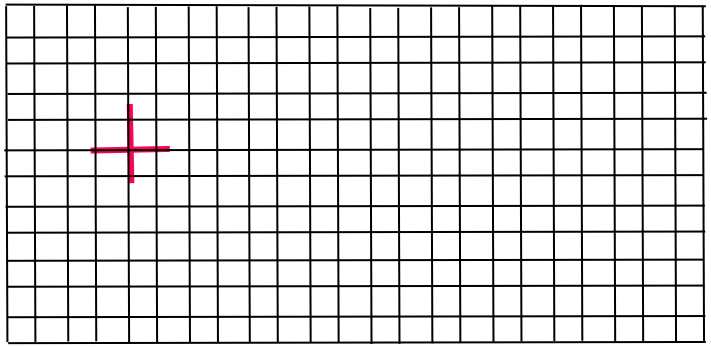
Sandia National Laboratories

BSP + msg agg
Eg multi-material shock solid mechanics

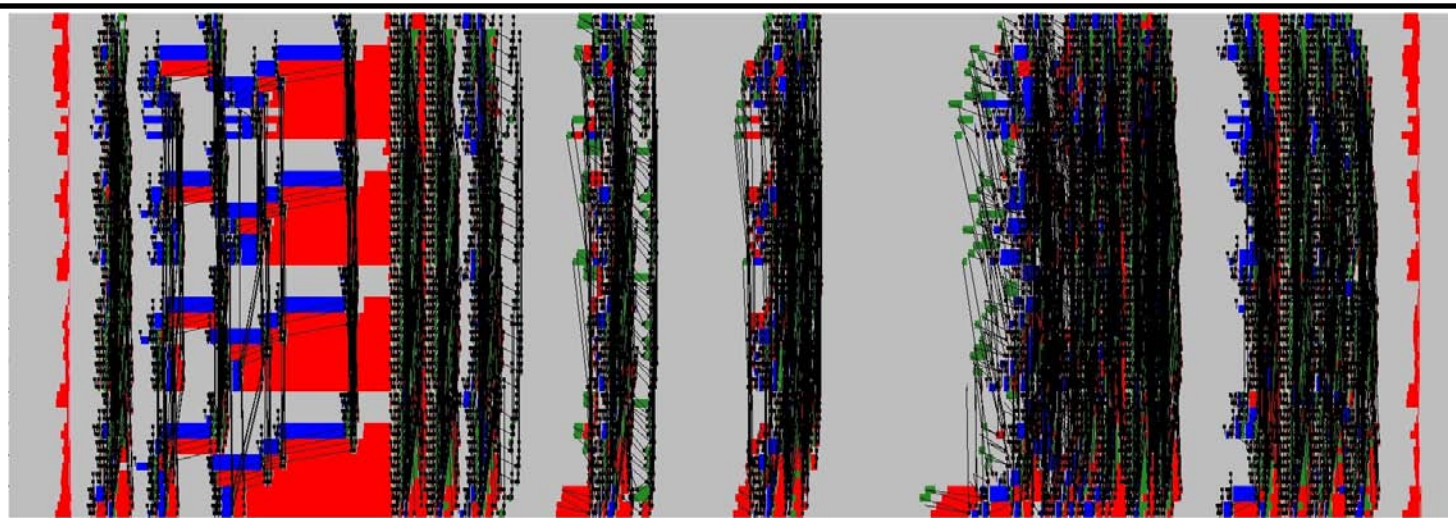
DO I = 1, NUM_VARS

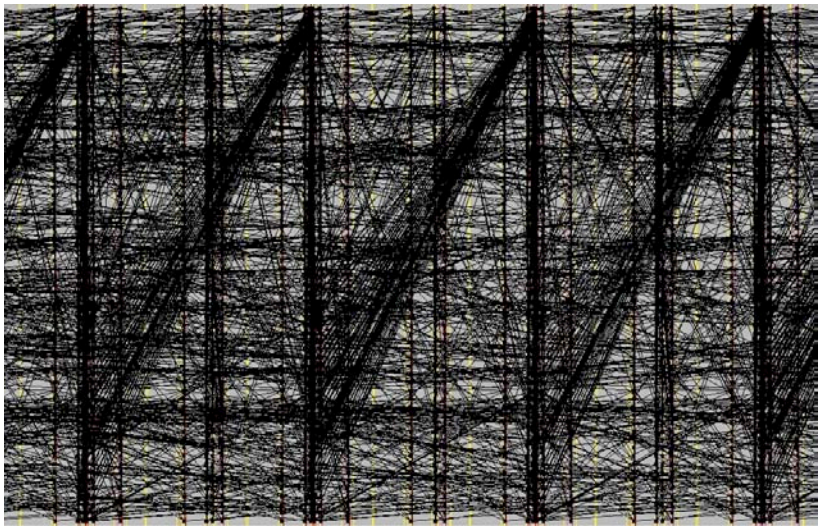


END DO
DO I = 1, NUM_VARS



END
DO





Sandia National Laboratories

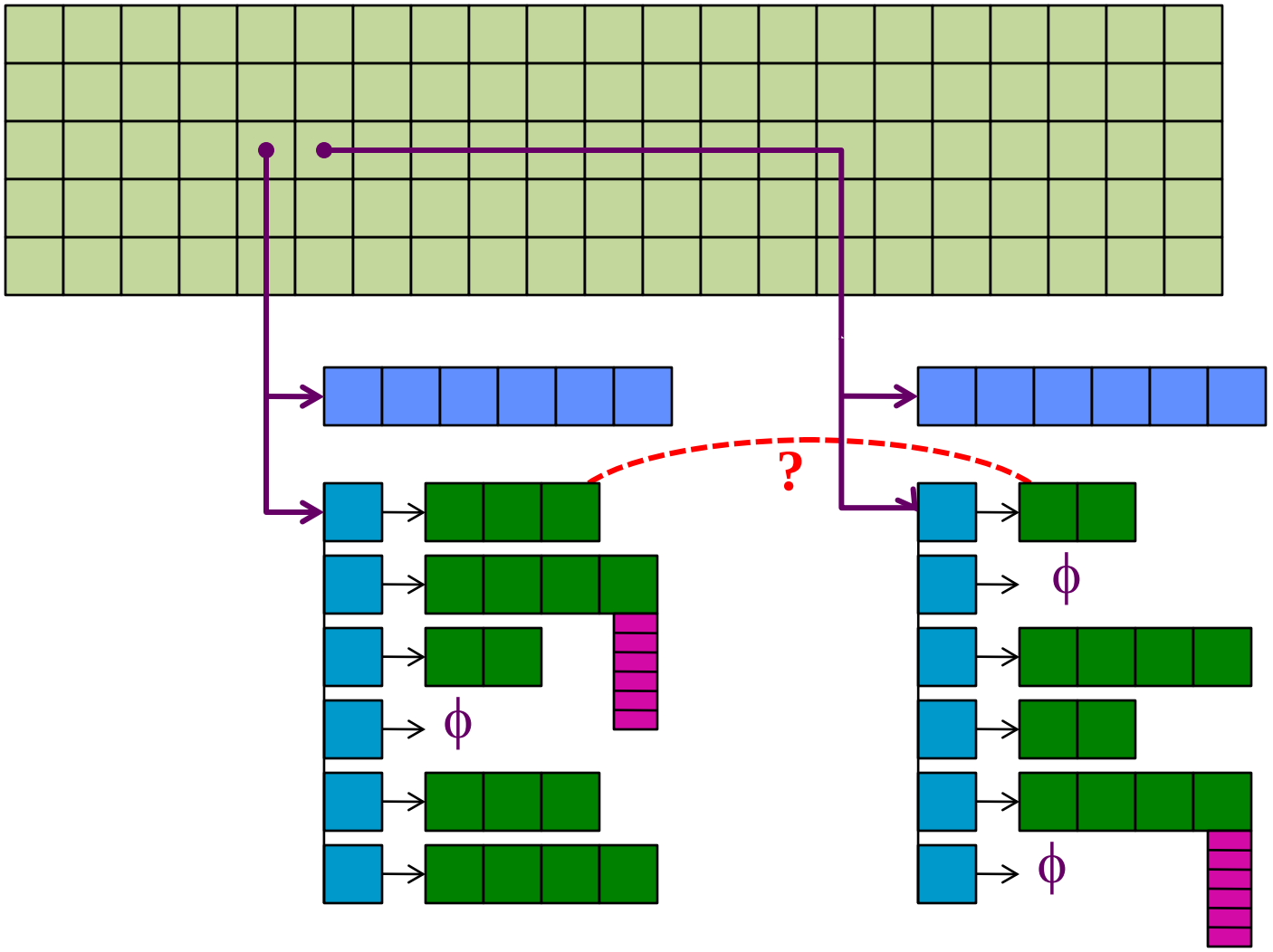
Dominant Issue

A million lines of code like this:

$$A(B(I)) = C(D(I))$$

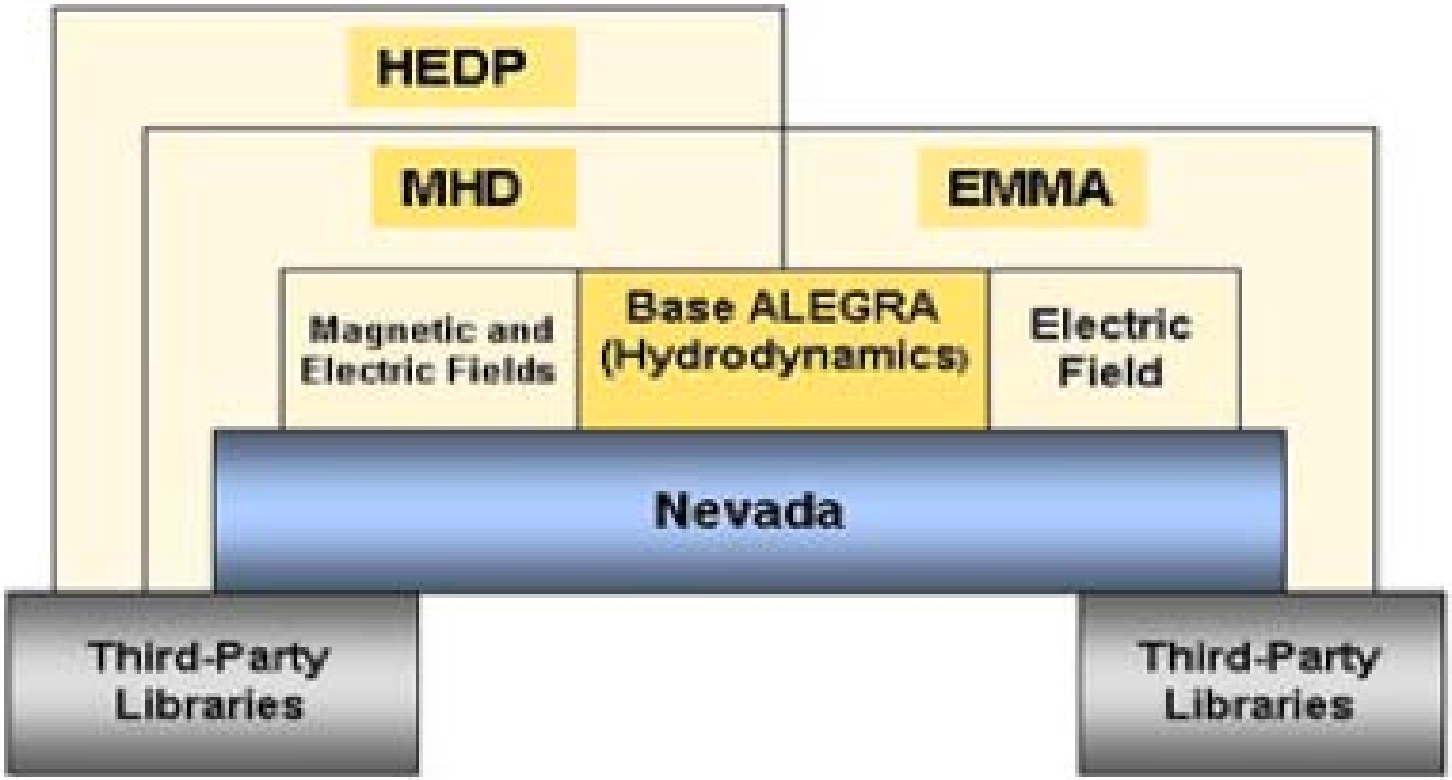
Sandia National Laboratories

Nice way to manage unstructured mesh



Sandia National Laboratories

A million lines of code is not created equally…



Whatever it is, I want:

- **Asynchronous movement of data between distributed memory processes,**
- **effective movement of non-contiguous data, and**
- **logical-to-physical map (locality controls).**

Summary

- Architectures in flux (but converging?)
- Programming mechanisms in flux (but converging?)
- Revolutionary code re-write a huge undertaking
- Not a computer science exercise (but publications are to be had)
- Science and engineering trust must be maintained throughout

$$A(B(I)) = C(D(I))$$

Sandia National Laboratories

Acknowledgements

- Sandia CSRF
- NNSA ASC CSSE

Sandia National Laboratories

Thanks

Sandia National Laboratories

Extra slides

ALEGRA code base*
(project began 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

Programming Model of the Future *(prediction, not a preference)*

- **SPMD MPI between nodes**
- **On-node: multiple "views" of the data structure; eg SIMD, SIMT, MIMD.**
- **C/C++/Fortran**
  - **With "helper" syntax/semantics, mechanisms, & libraries**

So said I, 8 June 2011, and again July 27, 2011.

Programming Model of the Future
(preference, not a prediction)

```
const
   PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n],
   AllSpace = PhysicalSpace.expand(1);

var
   Coeff, X, Y : [AllSpace] : real;

var
   Stencil = [ -1..1, -1..1 ];

forall i in PhysicalSpace do

   Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

Programming Model of the Future
(preference, not a prediction)

```
const
   DensPhysSpace: domain(2) distributed(Block) = [1..m, 1..n],
   AllSpace = PhysicalSpace.expand(1),
   SparseSpace = sparse subdomain ( AllSpace );

var
   Coeff, X, Y : [SparSpace] : real;

var
   Stencil = [ -1..1, -1..1 ];

forall i in SparseSpace do

   Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

Will the next programming model be an incremental change or a revolutionary change?

Yes.

It will (mostly) be what we should have been doing (and wanted to do) with SCOTS.

Like early days of message passing, will probably require evolutionary changes wrt programming mechanisms (eg CUDA, OpenCL, HMPP, PGI accel, XYZ, …, and MPI.)

Do we need to completely rethink our applications or will incremental approaches suffice?

Perhaps will inspire new algorithms/applications?