HPCS Languages : Potential for Scientific Computing

Richard Barrett
Application Performance Tools group
Oak Ridge National Laboratory

LACSS 2008
Santa Fe, NM
October 15, 2008

Managed by UT-Battelle for the
U. S. Department of Energy

OAK RIDGE National Laboratory

Productivity : What was the Question?

Richard Barrett
Application Performance Tools group
Oak Ridge National Laboratory

LACSS 2008
Santa Fe, NM
October 15, 2008

Managed by UT-Battelle for the
U. S. Department of Energy

OAK RIDGE National Laboratory

High Productivity Computing Systems (HPCS) Program

- DARPA initiated, 2004.
- "Workshop on Programming Languages for High Performance Computing (HPCWPL) Final Report", 2004.
- Three new languages :

Language	Spec	Implementation	
Chapel	v0.775	v0.7 compiler	Cray
Fortress	v1.0	v1.0 interpreter	Sun
X10	v1.7	v1.5(?) compiler	IBM

"Due" in 2011.

Productivity

"Time for idea to solution."

- DARPA

Characteristics of a Productive Programming Language

- Programmability
- Performance
- Portability
- Robustness

"Expressiveness"

(Barrett's list (and others))

How to mess this up

- Productivity is a synonym for programmability

  …and programmability is nebulous.

- We need a metric!

  (I've never heard a program manager ask for this.)

  - What's the productivity of Fortran? C? C++?
  - Now add parallelism : MPI, OpenMP, etc.

Measuring Performance
How to mess this up cont'd

- Take code someone else wrote,
- compile it using a compiler someone else wrote,
- run it on a machine someone else built,
- plop "metrics" into an Excel spreadsheet,
- submit to HPC conference;
- QED. (And luckily not reproducible.)

Early Evaluation of IBM Blue Gene/P, S. Alam, R. Barrett, M. Bast, M. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J.S. Vetter, P. Worley, W. Yu, Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC08), Austin, TX, 2008.

Measuring Performance
How to mess this up cont'd

- Take code someone else wrote,
- compile it using a compiler someone else wrote,
- run it on a machine someone else built,
- plop "metrics" into an Excel spreadsheet,
- submit to HPC conference;
- QED. (And luckily not reproducible.)

Cray X1 Evaluation Status Report, Agarwal, P.A., Alexander, R.A., Apra, E., Balay, S., Bland, A.S., Colgan, J., D'Azevedo, E.F., Dongarra, J., Drake, J. B., Dunigan, T.H., Dunning, T.H., Fahey, M.R., Fahey, R.A., Geist, A., Gorda, B., Gordon, M., Gropp, W. D., Harrison, R. J., Kendall, R., Keyes, D., Kaushik, D., Krishnakumar, M., Luszczek, P., Mezzacappa, A., Nichols, J.A., Nieplocha, J., Oliker, L., Packwood, T., Pindzola, M.S., Schulthess, T.C., Simon, H., Stevens, R., Vetter, J.S., White, J.B., Windus, T.L., Worley, P.H., Zacharia, T. (2004), ORNL/TM-2004/13.

Measuring Performance
How to mess this up cont'd

- Take code someone else wrote,
- compile it using a compiler someone else wrote,
- run it on a machine someone else built,
- plop "metrics" into a spreadsheet,
- submit the difference;
- QED. (And luckily not reproducible.)

Performance Potential?

Cray X1 Evaluation Status Report, Agarwal, P.A., Alexander, R.A., Apra, E., Balay, S., Bland, A.S., Colgan, J., D'Azevedo, E.F., Dongarra, J., Drake, J. B., Dunigan, T.H., Dunning, T.H., Fahey, M.R., Fahey, R.A., Geist, A., Gorda, B., Gordon, M., Gropp, W. D., Harrison, R. J., Kendall, R., Keyes, D., Kaushik, D., Krishnakumar, M., Luszczek, P., Mezzacappa, A., Nichols, J.A., Nieplocha, J., Oliker, L., Packwood, T., Pindzola, M.S., Schulthess, T.C., Simon, H., Stevens, R., Vetter, J.S., White, J.B., Windus, T.L., Worley, P.H., Zacharia, T. (2004), ORNL/TM-2004/13.

We want a MATLAB-like language

(How to mess this up cont'd)

Parallel is harder than serial,

and we don't have this in serial.

Example :

Verify that the "Betweenness Centrality" of a graph is positive (for all non-isolated vertices)

```
if ((BCmat >= 0) .* (BCmat <= numV*(numV-1))) > 0
```

We want a MATLAB-like language

Breaking news!

(ap cont'd)

We are beginning an assessment

Star-P

From Star-P web site, wrt MD sim.

My working assumptions

Parallel is harder than serial,

MPI is a very good thing,

Fortran is a good language, and

computational scientists are (always) looking for a better way, but…

they don't trust computer scientists, and

they are from Missouri…

ORNL is Preparing to "Make the Leap"

"Exploring HPCS Languages in Scientific Computing"

- David Bernholdt, Wael Elwasif, Aniruddha Shet (CS Research)
- Robert Harrison, (Comp Chem group)
- Valmor de Almeida, (Reactor Analysis Group)
- Richard Barrett, Jeffery Kuehn (Scientific Computing)
- Sadaf Alam (Future Technologies)
- Steve Poole (Chief Sc, Dir of Special Projects@CSMD)

- LDRD : "Preparing for New Programming Languages for Ultrascale Applications"

- Collaborators DoD / DARPA-HPCS Mission Partners

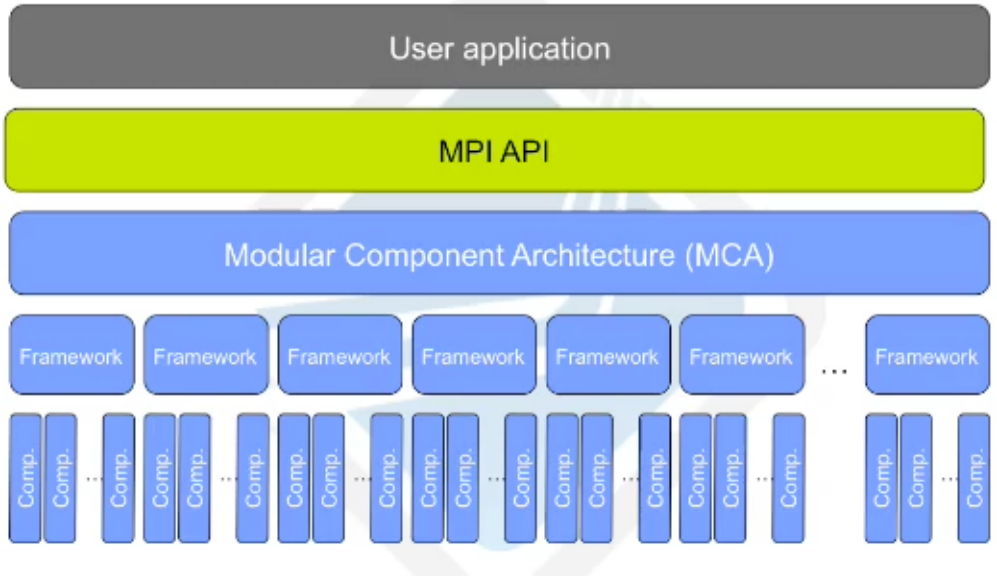
Why participate?

- The external quest for "a better way".
- Influencing language development
- Accelerate adoption
- How can we "think" in these languages?
- And what is the performance ***potential***?

Outline

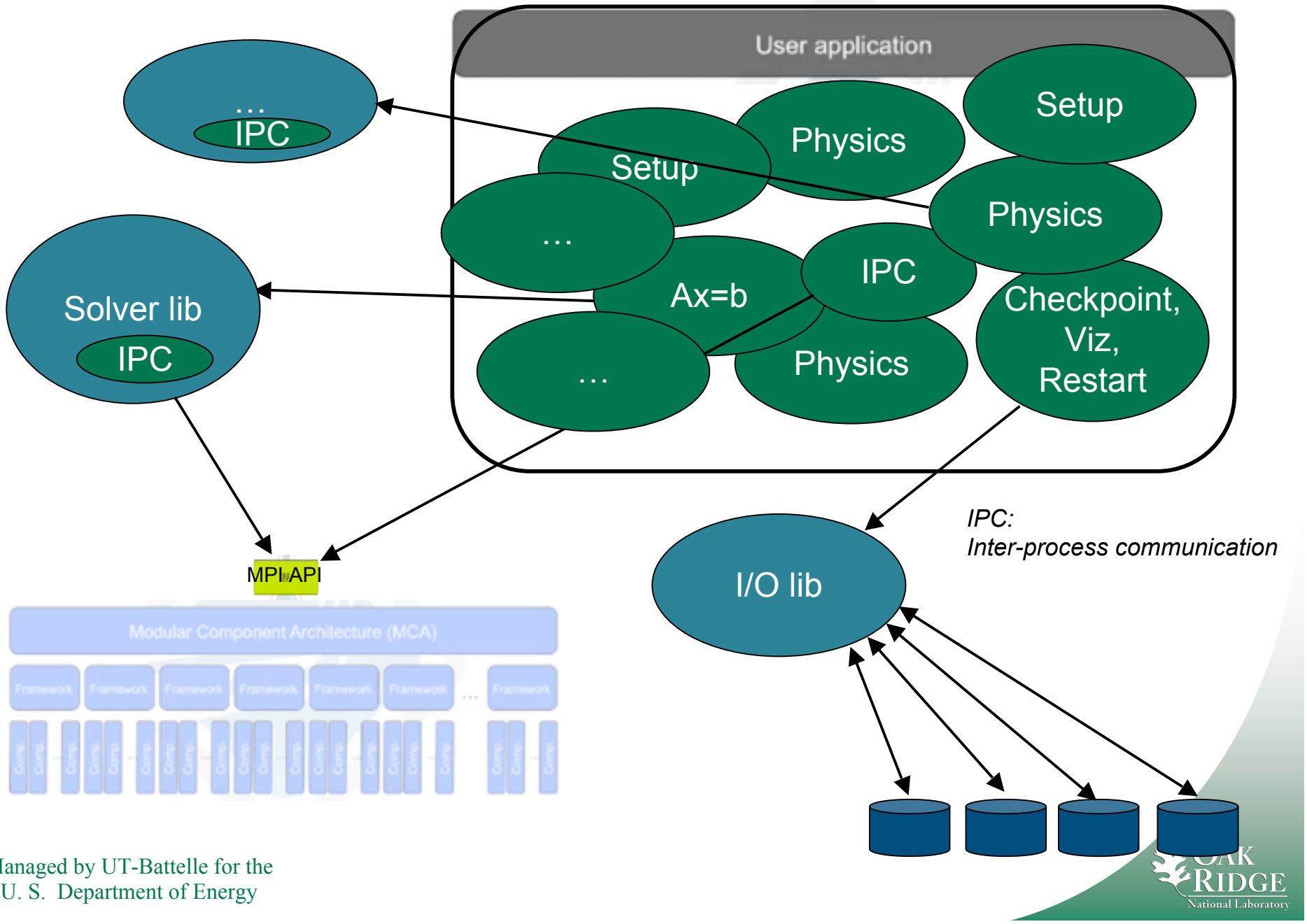
- Motivation
- Language overviews
- Case studies : let's look at some code!
- Some issues…

MPI developer* view of the Universe



** Image from openmpi.org*

Code developer view of the Universe



Application MPI use

Code	*Logical SLOC*	*MPI* pt2pt	coll	*All*
AORSA	20,671	2	0	3
GYRO	44k	2	4	17
HYCOM	18,533	3	3	18
MCNP	>100k	4	5	21
Moldt	767	0	0	0
NPB 3.3	20,671	4	5	33
POP(1.4.3)	16,770	4	3	23
POP(2.0.1)	21,304	4	4	17
ROMS	110,984	4	6	17
S3D	19,483	3	7	22
SAGE	109,000	5	6	37
sPPM	3,752	2	1	10
Sweep3d	1,081	2	3	9
Totals		8	9	80

Languages Overview

- Higher-level core language
  - Rich array data types, object oriented, generic programming, library-oriented, extensible
- Integrated concurrency
  - Task and data parallelism, multi-level concurrency, parallel loops/generators/iterators, atomic sections, futures, etc.
- Global view of data
- Backed by significant DARPA HPCS and vendor investment
- If not *the* future choice, then *representative* of it

Examples

- Difference stencils
- Hartree-Fock method
- Grid sweeping

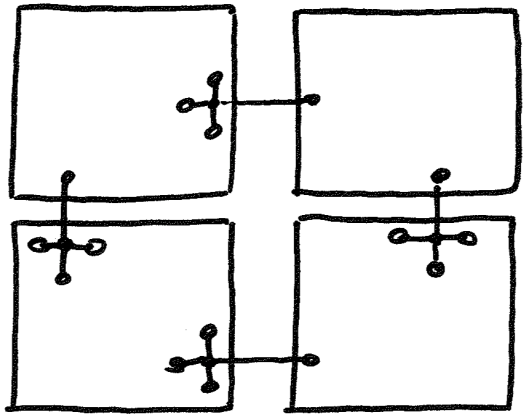
9-pt stencil : Fortran/MPI

```
CALL BOUNDARY_EXCHANGE ( ... )

DO J = 2, LCOLS+1
   DO I = 2, LROWS+1

      GRID2(I,J) =
           GRID1(I-1,J-1) + GRID1(I-1,J) + GRID1(I-1,J+1) +
           GRID1(I,J-1)   +  GRID1(I,J)  + GRID1(I,J+1)   +
           GRID1(I+1,J-1) + GRID1(I+1,J) + GRID1(I+1,J+1)
              / 9.0
   END DO
END DO
```

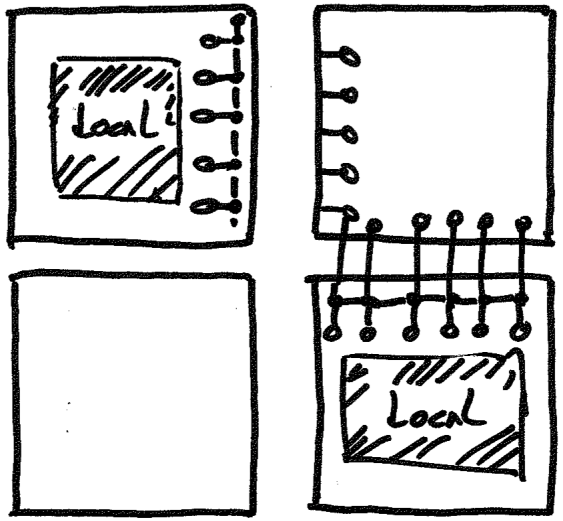
Many ways to share boundary data

Halo exchange

- many MPI functions, shmem, other.



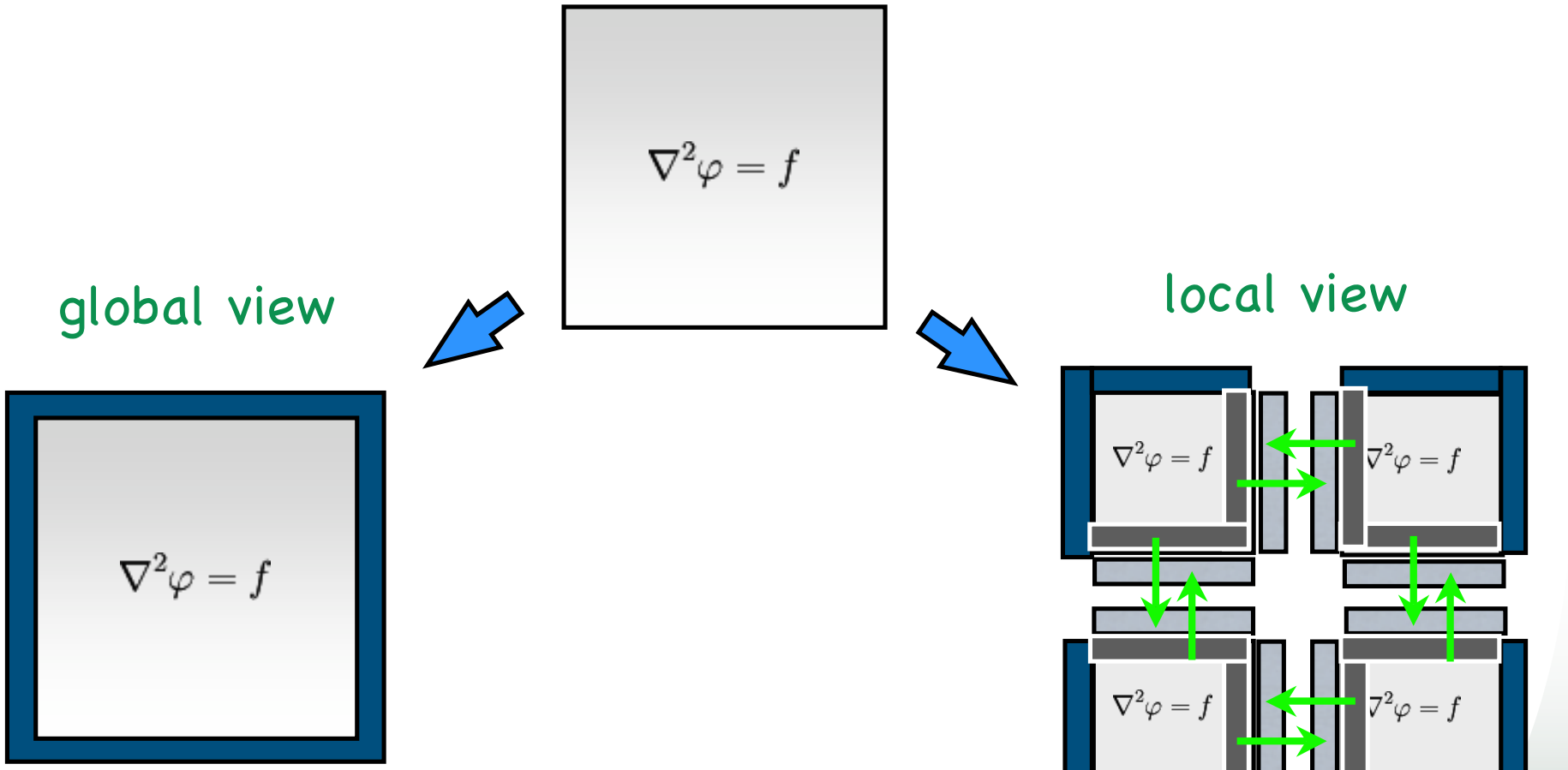
Load-it-when-you-need-it



Boundary sweep

- Many architecture features to exploit.

Global v Local view :



Chapel 9-pt stencil

Parallelism

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

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

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


forall i in PhysicalSpace do
   Y(i) = ( + reduce  [k in Stencil]  X (i+k) * div );
```

Fortress 5-pt stencil

```
stfivept [\Elt extends

  Number\](z:Elt,o:Elt) :() = do
  a = array[\Elt\](N,N)
  b = array[\Elt\](N,N)
  for j <- 2#(N-3) do
    for i <- 2#(N-3) do
      b[i,j] :=
      ( a[i-1,j]+a[i,j-1]+a[i,j]+a[i,j+1]+a[i+1,j] ) * div
    end
  end
```

X10 5-pt stencil

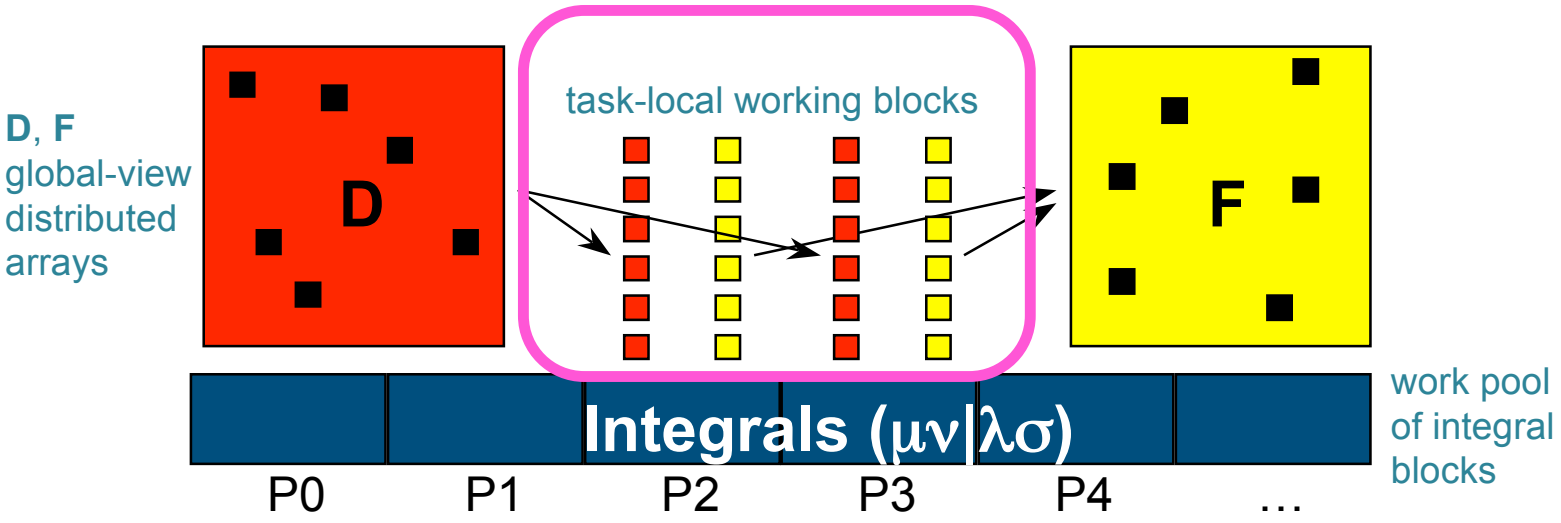
```
public class stencil {
  static final region (:rank==2) Rall = [0:N+1,0:N+1], Rcore = [1:N,1:N];
  static final dist    (:rank==2) Dall=
       (dist(:rank==2))dist.factory.block(Rall),
                                    Dcore=(Dall | Rcore),
                                    Dhalo=(Dall - Dcore.region);

final double[Dcore:rect&&rank == 2] gridnew =
        (double[Dcore:rect&&rank == 2]) new double[Dcore](point [i,j])
   {
          return ( (a[i-1,j]+a[i,j-1]+a[i,j]+a[i,j+1]+a[i+1,j] ) * div );
   };
```

Note: The new spec is based on Scala; syntax will change, but features will not.

Fock matrix build algorithm

Quantum chemistry problem from NWChem/Global Arrays

- scalable, irregular, global-view algorithm



Fock Matrix Build (1)
Language Features

Load balancing approach		Language constructs used		
		Chapel	Fortress	X10
Static, Program Managed		unstructured computations + locality control	explicit threads + locality control	asynchronous activities + locality control
Dynamic, Language (Runtime) Managed		iterators + forall loops	Multi-generator for loops	*not currently specified*
Dynamic, Program Managed	Task pool	synchronization variables	abortable atomic expressions	conditional atomic sections + futures
	Shared counter	synchronization variables	atomic expressions	unconditional atomic sections + futures

Fock Matrix Build (2)
Language Features Used

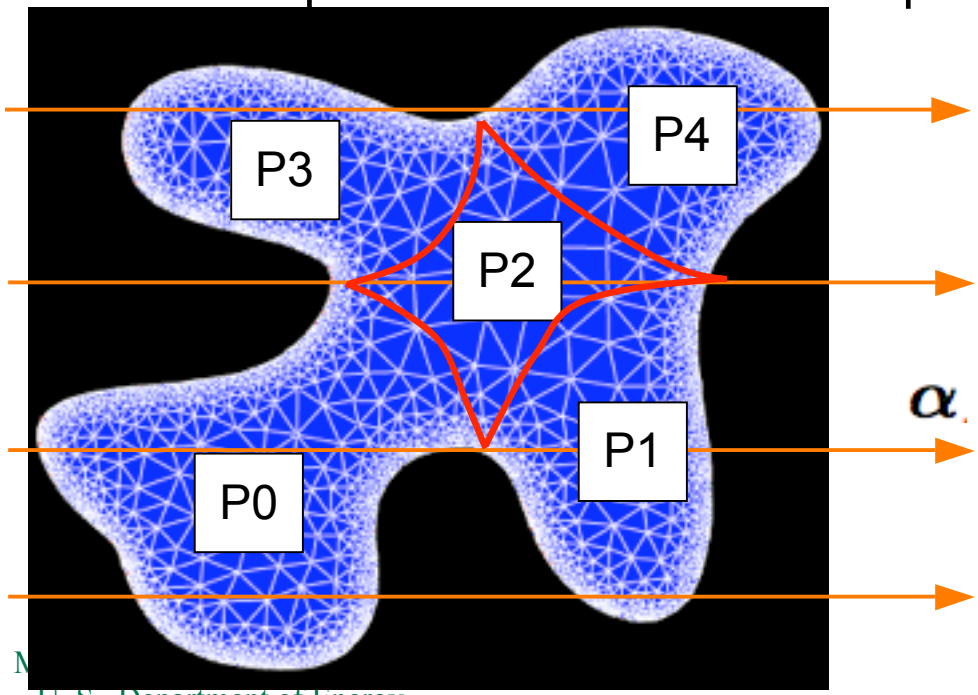
Operations		Language constructs used		
		Chapel	Fortress	X10
Mixed data and task parallelism		cobegin (task) + domain iterator (data)	tuple (task) + for loop (data)	finish async (task) + ateach (data)
Global-view array operations	initialization	array initialization expressions	comprehensions / function expressions	array initialization functions
	arithmetic	array promotions of scalar operators (+,*)	fortress library operators (+,juxtaposition)	array class methods (add,scale)
	sub-array	slicing	array factory functions (subarray)	restriction

Parallel Mesh Sweeping

- **Linear hyperbolic problems**
  - Linear characteristics (prototype)

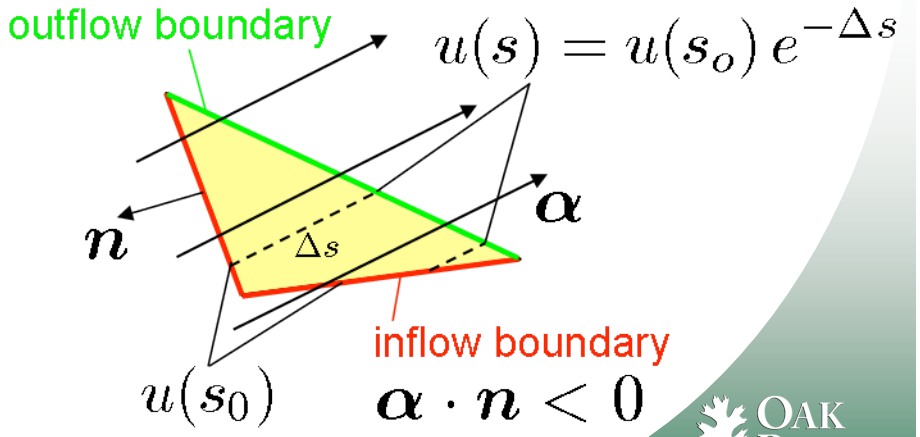
$$\nabla_{\boldsymbol{x}}\, u \cdot \boldsymbol{\alpha}(\boldsymbol{x}) + u = 0 \quad \text{in } \Omega$$

- **Linear algebra equivalent**
  - Finding independent sets of equations when corresponding matrix of coefficients is sparse
- **Domain decomposition classical approach**
  - Graph partitioning requires tedious, error-prone, data manipulation from user/developer



$$\frac{\mathrm{d}u}{\mathrm{d}s} = -u(s) \quad \text{on C}$$

$$\mathrm{C} := \{\boldsymbol{x} \in \Omega \mid \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \boldsymbol{\alpha}\}$$



Mesh Data Distribution (Chapel)

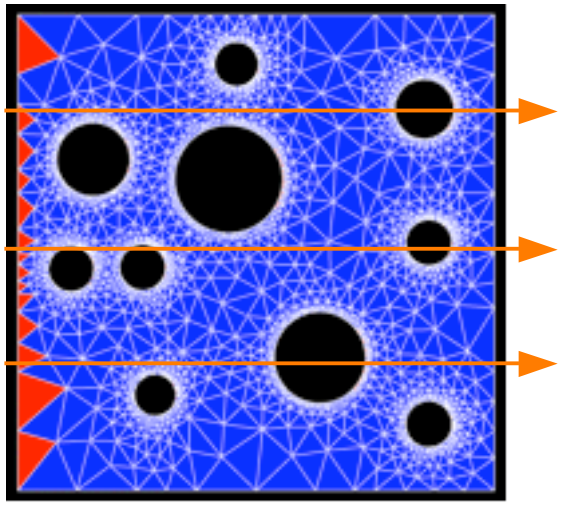
- **Distributed mesh data model**
  - **Generic type containers**
    - Nodes, edges, elements, regions
  - **Functionality**
    - I/O, dynamic allocation, search, traversal, garbage collection
- **Nested containers**
  - **FiniteElementPartition set**
    - members
    - nodesIds, nodesConnectivity
    - edgesIds, edgesConnectivity
    - nodesPositions
  - **FiniteElementSpaceMember**
    - geometry, nodesIds, edesIds
    - edgeBdrySides
  - **FiniteElementGeometry**
    - vertices, edges, edgesVertices…

```
class Vector { // generic/template class
// Accessors and modifiers
 def GetType() type {return T;}
 def GetSize() {return rng.high-rng.low+1;}
 def GetRange() {return rng;}
 def GetDomain() {return dom;}
 def Resize(n: range = 1..0) {rng = n; dom = [rng];}
 def Clear() {this.Resize();}
 def Find(val:T):bool {return LinearSearch(data,val)(1);}
 def these() var {for i in dom {yield data(i);}}
// Members
 type T;
 var rng:range(int);
 var value:T;
 var dom:domain(1) distributed(Block) = [rng];
 var data:[dom] T = value;
}
```

- ***Compiler distributes nested container data natively, avoiding explicit mesh partition and post-processing.***

Mesh Sweeping Implementation (Chapel)

- **Parallel execution**
  - **Multiple sweep directions**
  - **Traversal of containers**
    - elements and edges
  - **I/O**

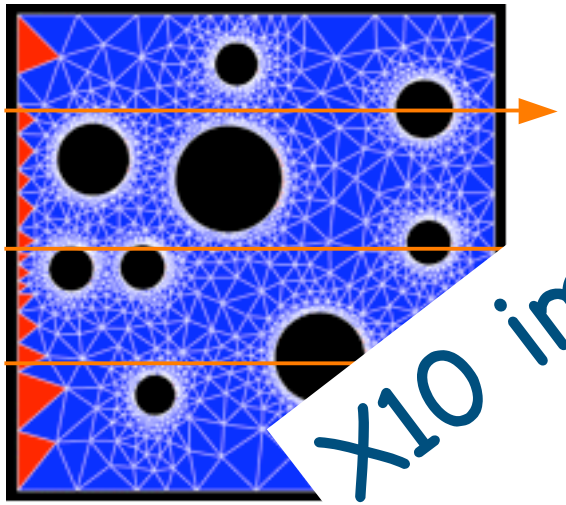


```
var setT = new FiniteElementPartition("mesh.dat");
const setTCardinality:int = setT.GetCardinality();
enum {blue, green, red};
forall sweepDir in sweepDirections {
 var waveFrontMask = new Vector(int,0..setTCardinality,blue);
 waveFrontMask(0) = green;
 while (waveFrontMask.Find(blue)) {
  forall fE in setT.GetMembers {
   if (waveFrontMask(fE.GetId()) == blue) {
    var setColor:bool = true;
    forall edge in fE.GetEdges() {
     const normal = fE.GetEdgeNormal(edge);
     if (normal^sweepDir < 0.0){
      const neMemberId:int = setT.GetNeighborMemberId(fE,edge);
      if (waveFrontMask(neMemberId) != green) {setColor = false;}
     }
     if (setColor == true) {waveFrontMask(fE.GetId()) = red;}
    }
   }
  }
  forall i in waveFrontMask {if (i != blue) {i = green;}}
 }
}
```

- ***Language allows for parallel execution of typical object-oriented code with virtually no changes.***

Mesh Sweeping Implementation (Chapel)

- **Parallel execution**
  - **Multiple sweep directions**
  - **Traversal of containers**
    - elements and edges
  - **I/O**



```
var setT = new FiniteElementPartition("mesh.dat");
const setTCardinality:int = setT.GetCardinality();
enum {blue, green, red};
forall sweepDir in sweepD      ns {
 var waveFrontMask = n          t,0..setTCardinality,blue);
 waveFrontMask(0) =
 while (waveFron
  forall fE i
   if (wav                      lue) {
    var
                                ) {
                                geNormal(edge);
                                0.0){
                                .int = setT.GetNeighborMemberId(fE,edge);
                                Mask(neMemberId) != green) {setColor = false;}

                                olor == true) {waveFrontMask(fE.GetId()) = red;}


  forall i in waveFrontMask {if (i != blue) {i = green;}}
 }
}
```

X10 implementation underway

- ***Language allows for parallel execution of typical object-oriented code with virtually no changes.***

Language Issues

- Type support and system (checking)
- Multi-dimensional arrays
- Language inter-operability (Global and local view)
- "Eureka" moment
- I/O
- Debugging / performance tuning
- Runtime management
- Paper to be released soon

Breaking out of a (parallel) loop:

Fortress:

```
label find1
  …
  parallel for
  if (some condition true/false) exit
end find1
```

- Cannot break from Chapel "forall"
- X10 can,

  but…
- What are actual semantic(s) and behavior?

Challenges for Acceptance of New Languages

HPF not accepted because

- immature compiler technology,
- lack of flexible distributions,
- inconsistent implementations,
- missing tools, and
- lack of patience by the community.

"The Rise and Fall of High Performance Fortran: an Historical Object Lesson", Kennedy, Koelbel and Zima, Proceedings of the Third ACM SIGPLAN conference on History of Programming Languages, 2007.

True motivation

8:20 AM Keynote Speaker: Dan Reed (Microsoft), *"The Future of Large-Scale Con*
9:50 AM Brian Albright (LANL), *"Application Design Considerations for Roadrunner Beyond"*
David Bader (George Institute of Technology), *"Accelerators, Cell Broadband Engine,Graphics Processors, and FPGAs"*
1:00 PM Peter Hofstee (IBM), *"The Case for Heterogeneous Multicore Processors*
1:40 PM Josep Torrellas (University of Illinois*), "Intrinsic Heterogenity in Multicores to Process Variation and Core Aging"*
2:20 PM Ken Koch (LANL), *Roadrunner: What Makes it Tick?"*
3:30 PM Steve Wallach (Convey), *"Computer Architecture: Past, Present,Future"*
4:10 PM Kevin Gildea (IBM), *"Petascale Challenges and Solutions"*
8:30 - 9:00 Mattan Erez (University of Texas at Austin*), "Parallelism isn't Enough: A Architect's Perspective on Building and Programming Terascale Processors and Petascale Systems"*
11:30 - 12:00 Peter Messmer (Tech-X), *"GPULib: GPU acceleration of scientific ap in high-level languages"*
8:30 - 9:00 John T. Daly (LANL*), "Resilience: Sacrificing Previous Convictions About Physical Laws"*
9:00 - 9:30 Garth Gibson (Carnegie Mellon University / Panasas, Inc*), "Failure in Supercomputers and Supercomputer Storage"*

Additional Information

Chapel	:	http://chapel.cs.washington.edu
Fortress	:	http://projectfortress.sun.com/Projects/Community
X10	:	http://x10-lang.org/

- *SC08 tutorials*
- *ORNL booth*

Further reading

"Exploring HPCS Languages in Scientific Computing", R.F. Barrett, S.R. Alam, V. de Almeida, D.E. Bernholdt, W.R. Elwasif, J.A. Kuehn, S.W. Poole, and A.G. Shet, Scientific Discovery Through Advantaced Computing (SciDAC'08), Journal of Physics: Conference Series 125 012034, 2008.

"Exploring the Performance Potential of Chapel", Barrett, Alam, and Poole, Proc. 50th Cray User Group meeting, 2008.

"Programmability of the HPCS Languages: A Case Study with a Quantum Chemistry Kernel", Shet, Elwasif, Harrison, and Bernholdt, HIPS'08, 2008.

"Expressing POP with a Global View Using Chapel: Toward a More Productive Ocean Model", Barrett, Alam, and Poole, ORNL Technical Report TM-2007/122, 2007.

"Finite Difference Stencils Implemented Using Chapel", Barrett, Roth, and Poole, ORNL Technical Report TM-2007/119, 2007.

"Strategies for Solving Linear Systems of Equations Using Chapel", Barrett and Poole, Proc. 49th Cray User Group meeting, 2007.

"Is MPI Hard? An Application Survey", SciComp group & others. submitted.

"HPLS: Preparing for New Programming Languages for Ultra-scale Applications", ORNL LDRD: Bernholdt, Barrett, de Almeida, Elwasif, Harrison, and Shet.

"Co-Array Fortran Experiences Solving PDE Using Finite Differencing Schemes", Barrett, Proc. 48th Cray User Group, 2006.

"UPC on the Cray X1E", Barrett, El-Ghazawi, Yao, 48th Cray User Group, 2006.

Acknowledgments

- Language development teams.

- This work has been supported by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), and the ORNL Postmasters Research Participation Program which is sponsored by ORNL and administered jointly by ORNL and by the Oak Ridge Institute for Science and Education (ORISE). ORNL is managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05- 00OR22725. ORISE is managed by Oak Ridge Associated Universities for the U. S. Department of Energy under Contract No. DE-AC05-00OR22750.