HPCS Languages: Potential for Scientific Computing

Richard Barrett
Application Performance Tools group
Oak Ridge National Laboratory

LACSS 2008
Santa Fe, NM
October 15, 2008
Productivity: What was the Question?

Richard Barrett
Application Performance Tools group
Oak Ridge National Laboratory

LACSS 2008
Santa Fe, NM
October 15, 2008
High Productivity Computing Systems (HPCS) Program

- Three new languages:

<table>
<thead>
<tr>
<th>Language</th>
<th>Spec</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapel</td>
<td>v0.775</td>
<td>v0.7 compiler</td>
</tr>
<tr>
<td>Fortress</td>
<td>v1.0</td>
<td>v1.0 interpreter</td>
</tr>
<tr>
<td>X10</td>
<td>v1.7</td>
<td>v1.5(?) compiler</td>
</tr>
</tbody>
</table>

“Due” in 2011.
Productivity

“Time for idea to solution.”

- DARPA

Characteristics of a Productive Programming Language

• Programmability
• Performance
• Portability
• Robustness

(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.)

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.)

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.)

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)

\[
\text{if } ((\text{BCmat} \geq 0) \times (\text{BCmat} \leq \text{numV} \times (\text{numV}-1))) > 0
\]
We want a MATLAB-like language

**Breaking news!**

We are beginning an assessment

Star-P

Eliminated 6 Months of C & MPI Programming

*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

User application

- Setup
- Physics
- IPC
- Checkpoint, Viz, Restart

Solver lib
- IPC
- Ax=b

I/O lib
- Inter-process communication

MPI API

Managed by UT-Battelle for the U.S. Department of Energy
# Application MPI use

<table>
<thead>
<tr>
<th>Code</th>
<th>Logical SLOC</th>
<th>MPI pt2pt</th>
<th>MPI coll</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>AORSA</td>
<td>20,671</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>GYRO</td>
<td>44k</td>
<td>2</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>HYCOM</td>
<td>18,533</td>
<td>3</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>MCNP</td>
<td>&gt;100k</td>
<td>4</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>Moltdt</td>
<td>767</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NPB 3.3</td>
<td>20,671</td>
<td>4</td>
<td>5</td>
<td>33</td>
</tr>
<tr>
<td>POP(1.4.3)</td>
<td>16,770</td>
<td>4</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>POP(2.0.1)</td>
<td>21,304</td>
<td>4</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>ROMS</td>
<td>110,984</td>
<td>4</td>
<td>6</td>
<td>17</td>
</tr>
<tr>
<td>S3D</td>
<td>19,483</td>
<td>3</td>
<td>7</td>
<td>22</td>
</tr>
<tr>
<td>SAGE</td>
<td>109,000</td>
<td>5</td>
<td>6</td>
<td>37</td>
</tr>
<tr>
<td>sPPM</td>
<td>3,752</td>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Sweep3d</td>
<td>1,081</td>
<td>2</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>8</strong></td>
<td><strong>9</strong></td>
<td></td>
<td><strong>80</strong></td>
</tr>
</tbody>
</table>
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:

$\nabla^2 \varphi = f$

global view

local view
Chapel 9-pt stencil

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 );
stfivept [\Elt extends

\text{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
end
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 );
    };
Fock matrix build algorithm

Quantum chemistry problem from NWChem/Global Arrays

• scalable, irregular, global-view algorithm
## Fock Matrix Build (1)

**Language Features**

<table>
<thead>
<tr>
<th>Load balancing approach</th>
<th>Language constructs used</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chapel</strong></td>
<td><strong>Fortress</strong></td>
</tr>
<tr>
<td>Static, Program Managed</td>
<td>unstructured computations + locality control</td>
</tr>
<tr>
<td>Dynamic, Language (Runtime) Managed</td>
<td>iterators + forall loops</td>
</tr>
<tr>
<td>Dynamic, Program Managed</td>
<td>synchronization variables</td>
</tr>
<tr>
<td>Task pool</td>
<td>Multi-generator for loops</td>
</tr>
<tr>
<td>Shared counter</td>
<td>synchronization variables</td>
</tr>
<tr>
<td></td>
<td>atomic expressions</td>
</tr>
</tbody>
</table>

Managed by UT-Battelle for the U. S. Department of Energy
# Fock Matrix Build (2)

## Language Features Used

<table>
<thead>
<tr>
<th>Operations</th>
<th>Language constructs used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapel</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Mixed data and task</td>
<td>cobegin (task)</td>
</tr>
<tr>
<td>parallelism</td>
<td>+ domain iterator (data)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Global-view array operations</td>
<td>initialization</td>
</tr>
<tr>
<td></td>
<td>array initialization</td>
</tr>
<tr>
<td></td>
<td>expressions</td>
</tr>
<tr>
<td>arithmetic</td>
<td>array promotions of</td>
</tr>
<tr>
<td></td>
<td>scalar operators (+,*)</td>
</tr>
<tr>
<td>sub-array</td>
<td>slicing</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parallel Mesh Sweeping

- Linear hyperbolic problems
  - Linear characteristics (prototype)

- 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

\[ \nabla_x u \cdot \alpha(x) + u = 0 \quad \text{in } \Omega \]

\[ \frac{du}{ds} = -u(s) \quad \text{on } C \]

\[ C := \{ x \in \Omega \mid \frac{dx}{ds} = \alpha \} \]

Outflow boundary:

\[ u(s) = u(s_0) e^{-\Delta s} \]

Inflow boundary:

\[ u(s_0) \quad \alpha \cdot n < 0 \]
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…

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

```plaintext
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;
}
```
Mesh Sweeping Implementation (Chapel)

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

```chapel
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;}
        }
      }
      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.**

Managed by UT-Battelle for the U. S. Department of Energy
Mesh Sweeping Implementation (Chapel)

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

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

```chapel
var setT = new FiniteElementPartition("mesh.dat");
const setTCardinality:int = setT.GetCardinality();
enum {blue, green, red};
forall sweepDir in sweepDirections {
    var waveFrontMask = new NeighborMemberIdSet(0..setTCardinality,blue);
waveFrontMask(0) = red;
    while (waveFrontMask) {
        forall fE in waveFrontMask {
            if (waveFrontMask(fE.Id()) == blue) {
                var edge = EdgeNormal(fE);
                if (edge.NormalAngle < 0.0) {
                    int neMemberId = 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;}}
    }
}
```

X10 implementation underway
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.

True motivation

8:20 AM Keynote Speaker: Dan Reed (Microsoft), “The Future of Large-Scale Computing”
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 Heterogeneity in Multicores to Process Variation and Core Aging”
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: An 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 applications 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


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


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.