Exploring the Performance Potential of Chapel

Richard Barrett, Sadaf Alam, and Stephen Poole

Scientific Computing Group
National Center for Computational Sciences

Future Technologies Group
Computer Science and Math Division

Oak Ridge National Laboratory
Chapel Status

• Compiler version 0.7, released April 15.
  • running on my Mac; also Linux, SunOS, CygWin
  • Initial release December 15, 2006.
  • End of summer release planned.
• Spec version 0.775
• Development team “optimally” responsive.
Productivity

Programmability

Performance

Portability

Robustness
Programmability:
Motivation for “expressiveness”

“By their training, the experts in iterative methods expect to collaborate with users. Indeed, the combination of user, numerical analyst, and iterative method can be incredibly effective. Of course, by the same token, inept use can make any iterative method not only slow but prone to failure. Gaussian elimination, in contrast, is a classical black box algorithm demanding no cooperation from the user.

Surely the moral of the story is not that iterative methods are dead, but that too little attention has been paid to the user's current needs?"

“Progress in Numerical Analysis”,
Beresford N. Parlett,
“Expressive” language constructs

Syntax and semantics that enable:

- algorithmic description
- provide intent to compiler & RTS

Cray User Group 2008, Helsinki
May 7, 2008
Prospective for Adoption

Must provide compelling reason

Performance

My view: Must exceed performance of MPI.

(Other communities may have different requirements.)

Rename “FORTRAN”
The Chapel Memory Model

There ain't one.
Finite difference solution of Poisson’s Eqn

\[ \nabla^2 \varphi = f \]

global view

local view

\[ \nabla^2 \varphi = f \]

\[ \nabla^2 \varphi = f \]
for i = 1, 2, ...
    solve \( Mz^{(i-1)} = r^{(i-1)} \)
    \( \rho_{i-1} = r^{(i-1)^T} z^{(i-1)} \)
    if ( i = 1 )
        \( p = z^{(0)} \)
    else
        \( \beta = \rho_{i-1} / \rho_{i-2} \)
        \( p = z^{(i-1)} + \beta p^{(i-1)} \)
    end if
    \( q = Ap \)
    \( \alpha = \rho_{i-1} / p^T q \)
    \( x = x^{(i-1)} + \alpha p \)
    \( r = r^{(i-1)} - \alpha q \)
    check convergence; continue if necessary
end
Linear equations may often be defined as "stencils" (Matvec, preconditioner)
CALL BOUNDARY_EXCHANGE (...)

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

    Y(I,J) =

    A(I-1,J-1) *X(I-1,J-1) + A(I-1,J) *X(I-1,J) + A(I-1,J+1) *X(I-1,J+1) +
    A(I,J-1)*X(I,J-1) + A(I,J)*X(I,J) + A(I,J+1) *X(I,J+1) +
    A(I+1,J-1) X(I+1,J-1) + A(I+1,J)*X(I+1,J) + A(I+1,J+1)*X(I+1,J+1)

  END DO
END DO
Co-Array Fortran implementations

IF ( NEIGHBORS(SOUTH) /= MY_IMAGE ) &
GRID1( LROWS+2, 2:LCOLS+1 ) = GRID1( 2,2:LCOLS+1 )[NEIGHBORS(SOUTH)]

Load-it-when-you-need-it

One-sided

Boundary sweep
Cray X1E
Heterogeneous, Multi-core

1024 Multi-streaming vector processors (MSP)

Each MSP
4 Single Streaming Processors (SSP)
4 scalar processors (400 MHz)
Memory bw is roughly half cache bw.
2 MB cache
18+ GFLOP peak

4 MSPs form a node
8 GB of shared memory.
Inter-node load/store across network. 56 cabinets
5-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

100x100 grid/pe
5-pt stencil; weak scaling

CAF: liwni
CAF: Segm
CAF: 1-sided
MPI

500x500 grid/pe
5-pt stencil; weak scaling

CAF: liwni
CAF: Segm
CAF: 1-sided
MPI

1kx1k grid/pe

X1E msp

gflops
Weak scaling performance

5-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

2kx2k grid/pe

GFLOPS

X1E msp
5-pt stencil; weak scaling

CAF
CAF: liwni
CAF: Segm
CAF: 1-sided
MPI

X1E msp

gflops

4kx4k grid/pe
Weak scaling performance

5-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

6kx6k grid/pe
5-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI
9-point stencil

CAF: four extra partners processes (corners)

MPI: same number of partners (with coordination)
9-pt stencil; weak scaling

CAF: liwnyi
CAF: Segm
CAF: 1-sided
MPI

500x500 grid/pe
9-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

1kx1k grid/pe

gflops vs. X1E msp
9-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI
9-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

4kx4k grid/pe

gflops

X1E msp
9-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

4kx4k grid/pe

gflops

X1E msp
9-pt stencil; weak scaling

CAF: liwnyi
CAF: Segm
CAF: 1-sided
MPI

6kx6k grid/pe

gflops

X1E msp
9-pt stencil; weak scaling

CAF: liwyni
CAF: Segm
CAF: 1-sided
MPI

8kx8k grid/pe

gflops

XIE msp
Chapel: Reduction implementation

```chapel
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));
```

Parallelism
Matrix as a “sparse domain” of 5 pt stencils

\textbf{const}
PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n],
AllSpace = PhysicalSpace.expanded(1);

\textbf{var}
Coeff, X, Y : [AllSpace] : real;

\textbf{var}
Stencil9pt = [ -1..1, -1..1 ],
Stencil = sparse subdomain (Stencil9pt) = [(i,j) in Stencil9pt]
if ( abs(i) + abs(j) < 2 ) then (i,j);

\textbf{forall} i in PhysicalSpace do

\quad Y(i) = ( + \textbf{reduce} [k in Stencil] Coeff (i+k) \times X (i+k) );
SN transport:
Exploiting the Global-View Model

Global-view

Local-view
SN transport:
Exploiting the Global-View Model


Cray User Group 2008, Helsinki
May 7, 2008
SN transport:
Exploiting the Chapel Memory Model

“$S_N$ Algorithm for the Massively Parallel CM-200 Computer”,
Randal S. Baker and Kenneth R. Koch, Los Alamos National Laboratory,

(t3d shmem version, too.)

Cray User Group 2008, Helsinki
May 7, 2008
AORSA arrays in Chapel

```
const

FourierSpace : domain(2) distributed (Block)

var
  fgrid,
  mask
  : [FourierSpace] real;

var
  PhysSpace: sparse subdomain (FourierSpace) =
  [i in FourierSpace] if mask(i) == 1 then i;

var
  pgrid
  : [PhysSpace] real;

ierr = pzgesv (..., PhysSpace);
  // ScaLAPACK routine
```

Dense linear solve, so interoperability needed.

**Fourier space**

**Real** space

Cray User Group 2008, Helsinki
May 7, 2008
Performance Expectations

1. If we had a compiler we could “know”.
2. “Domains” define data structures; coupled with operators.
3. Distribution options (including user defined)
4. Multi-Locales
5. Inter-process communication flexibility
6. Memory Model
7. Diversity of Architectures emerging
8. Strong funding model
Past, Current, and Future work

Acknowledgments

This research was sponsored by the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Chapel development team.

ORNL LDRD, DoD, AORSA project team.

SciDAC’08 program committee (Invited paper)