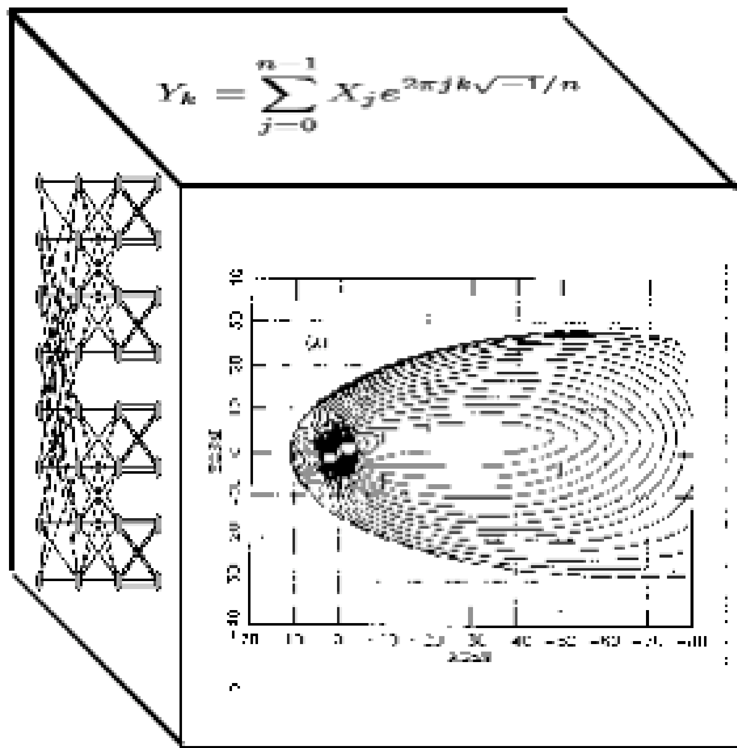
Parallelism in Fortress

Jan-Willem Maessen
Sun Microsystems
JanWillem.Maessen@sun.com

PGAS, 2005-09-14

Guy Steele
Jan-Willem Maessen
Eric Allen
David Chase
Victor Luchangco
Sam Tobin-Hochstadt
Sukyoung Ryu
Yossi Lev
Carl Eastlund
Joe Hallett
João Dias

Goal: Science-centered computation



- Program structure should reflect the science
- Not FLOPS
- Not communication structure
- Fortress is not there yet

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sun microsystems

Making Programming Easier

- Readability
- Conciseness
- Safety
- Programming in the large
- Parallelization with grace
- Simplicity of tool support
  - Parser and AST included for tool builders
- Domain-specific syntactic extension

- Build a language that's useful for more than just HPC

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sun microsystems

Generics, constraints, formatting

```
mmplus[/T extends Field/]
        (left : T[m BY n],
         right : T[n BY p],
         result : T[m BY p]) =
```

Constrained generic

Size constraints

$$mmplus[\![T \text{ extends } Field]\!]$$
$$(left : T_{m\times p},$$
$$right : T_{p\times n},$$
$$result : T_{m\times n}) =$$

Readable form

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sun microsystems

Components and APIs

```
api fortress.sparse
  import * from numeric
  import * from array
  trait Sparse⟦T extends Numeric, m×n⟧
      extends MatrixLike⟦T,m×n⟧
    mm(b : Sparse⟦T,n×p⟧) : Sparse⟦T,m×p⟧
    mvm(v : VectorLike⟦T,n⟧) : VectorLike⟦T,m⟧
  end
end
```

Declarations of functionality

```
component fortress.sparseMatrix
export fortress.sparse
  trait Sparse⟦T extends Numeric, m×n⟧
      extends MatrixLike⟦T,m×n⟧
    mm(b)
    mvm(v) = do
      r : T[m] = 0
      for e,i,j ← self.elements() do
        atomic r[i] += e * v[j]
      end
      r
    end
  end
end
```

API implemented

Types inferred

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sun microsystems

Fortress: A Trait-Based Language

- Objects are described by *traits* rather than classes
- Collection of related functionality, with optional default implementations
- Traits can be parametric

```
trait Sparse⟦T extends Numeric, m×n⟧
  extends MatrixLike⟦T,m×n⟧
  mm(b : Sparse⟦T,n×p⟧) : Sparse⟦T,m×p⟧
  mvm(v : VectorLike⟦T,n⟧) : VectorLike⟦T,m⟧ = do
    r : T[m] = 0
    for e,i,j ← self.elements() do
      atomic r[i] += e * v[j]
    end
    r
  end
end
```

Method defined by implementing objects

Default code for derived method

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sun microsystems

Operator definition

Goal: principled overloading (only use + for addition!)

```
trait Equality⟦T extends Equality⟦T⟧⟧
  equals(x:T):Boolean

trait Ordering⟦T extends Ordering⟦T⟧⟧
  extends Equality⟦T⟧
    le(x:T):Boolean
    gt(other:T) = not (self.le(other))
    ge(other:T) = other.le(self)
    lt(other:T) = not (self.ge(other))

opr =⟦T extends Equality⟦T⟧⟧(x:T, y:T):T = x.eq(y)
opr ≤⟦T extends Ordering⟦T⟧⟧(x:T, y:T):T = x.le(y)
opr <⟦T extends Ordering⟦T⟧⟧(x:T, y:T):T = x.lt(y)
opr >⟦T extends Ordering⟦T⟧⟧(x:T, y:T):T = x.gt(y)
opr ≥⟦T extends Ordering⟦T⟧⟧(x:T, y:T):T = x.ge(y)
```

Idiom for self types

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Machine Assumptions

- Programmers don't know the number of processors
  - Prototype small, run on a larger partition
  - Change partition size on the fly
- Computational load is always out of balance
- Need to expose vast amounts of parallelism
  - 10K CPUs × 50 threads each × slack for load balancing = many millions of simultaneous threads
  - Fine-grained threading is a must
- Programming model uses a shared address space
  - But distant access may be slow

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Parallelism Model

- The for loop is parallel by default
- Use recursive subdivision
  - > Scheduling selects appropriate granularity at run time
  - > Adapts to variations in machine size
- Distributions make subdivision easy
  - > Loops, arrays subdivide according to a distribution
  - > Distributions contain locality information for scheduling
- Transactional memory simplifies synchronization
  - > Still need efficient bulk operators (eg prefix)

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Transactional Memory

- Primitive synchronization mechanism in Fortress
- Code in an atomic block appears to execute in a single atomic step.

```
mvm(v) = do
  r : T[m] = 0
  for e,i,j ← self.elements() do
    atomic r[i] += e * v[j]
  end
  r
end
```

We update the vector atomically so that parallel updates are not lost.

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Data Parallelism

```
conjGrad⟦E extends Numeric⟧
        (A: Sparse⟦E, n×n⟧, x: E[n]): (E[n], E) = do
   cgitmax = 25
   z: E[n] := 0
   r: E[n] := x
   p: E[n] := r
   ρ: E := r·r
   for j ← sequential(0#cgitmax) do
      q = A p
      α = ρ / p·q
      z += α p
      r -= α q
      ρ0 = ρ
      ρ := r·r
      β = ρ / ρ0
      p := r + β p
   end
   (z, ‖x - A z‖)
end
```

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Adding a distribution

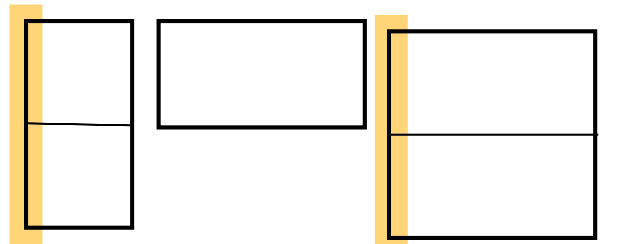
```
conjGrad⟦E extends Numeric⟧
         (A: Sparse⟦E, n×n⟧, x: E[n]): (E[n], E) = do
   cgitmax = 25
   z: E[n] := A.distribution.array⟦n⟧(0)
   r: E[n] := x.copy(distribution = A.distribution)
   p: E[n] := r.copy()
   ρ: E := r•r
   for j ← sequential(0#cgitmax) do
      q = A p
      α = ρ / p•q
      z := z + α p
      r := r – α q
      ρ0 = ρ
      ρ := r•r
      β = ρ / ρ0
      p := r + β p
   end
   (z, ‖x – A z‖)
end
```

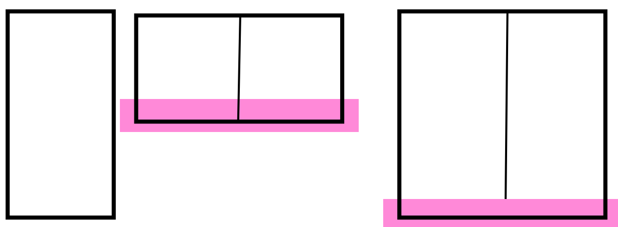
Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

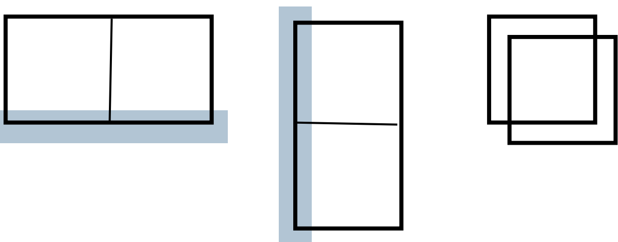
Cache oblivious matrix multiplication

```
mmplus⟦T extends Field⟧(A: T[m × n],
                        B: T[n × p],
                        result: T[m × p]):() =
case largest of
     1 ⇒ result[0,0] += A[0,0] B[0,0]

     m ⇒ [ Atop ; Abottom ] = A
         [ restop ; resbottom ] = result
         ( mmplus(Atop, B, restop),
           mmplus(Abottom, B, resbottom))
         ()

     p ⇒ [ Bleft Bright ] = B
         [ resleft resright ] = result
         ( mmplus(A, Bleft, resleft),
           mmplus(A, Bright, resright) )
         ()

     n ⇒ [ Aleft Aright ] = A
         [ Btop ; Bbottom ] = B
         (* IN ORDER *)
         mmplus(Aleft, Btop, result)
         mmplus(Aright, Bbottom, result)
 end
```

Tuple expressions run in parallel (fork/join)



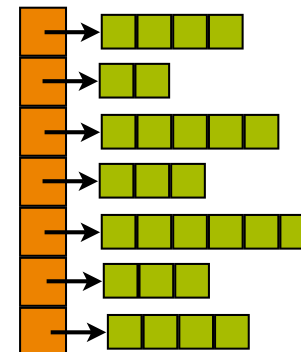




Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Distributions for Sparse Matrices

```
object SparseRow⟦T extends Numeric, n×m⟧
    (gen : Generator⟦T,m,n⟧,
     distribution : Distribution = rowMajor() )
    traits Sparse⟦T,m×n⟧
  ...
  r : (T[ ])[m] = distribution.array([ ])
  r[i] := distribution.shift(i).transpose.array(s),
      (s,i) ← rowSize.elements
    ...
end
```

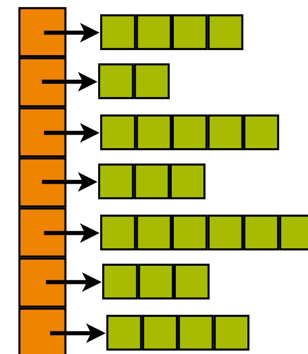


Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Sparse matrix-vector multiply

```
mvm(v) =
  [ i =
     SUM[(e,j)←row.elements] e.val v[e.pos] |
        (row, i) ← rows.indices ]
```

- Output vector follows row distribution
- Input vector is accessed randomly
- Can express replication of v by creating a 2-dimensional intermediate array
- Or, re-distribute v by copying...



Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Regions: Describing machine resources

- Regions describe hierarchical structure of machine (each node is a region):

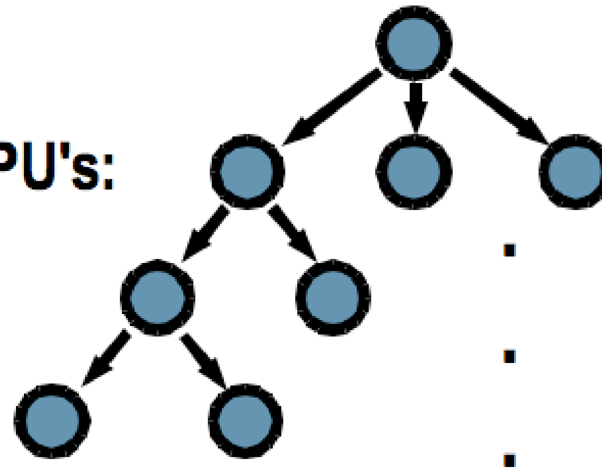
Entire machine:

Horizontal group of CPU's:

CPU's

:

Cores:



Tree is limited

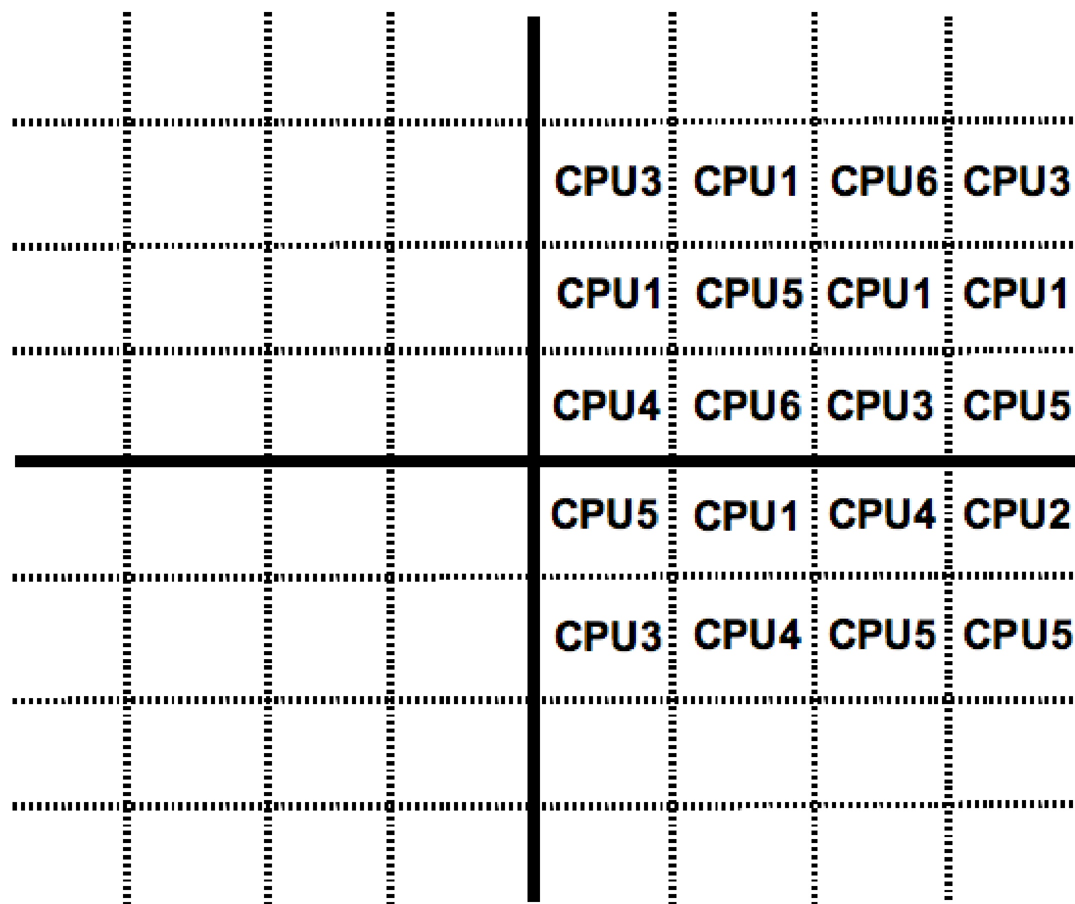
- Can not describe all architectures (e.g. grid layout)
- Support least upper bound

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Distributions: Allocating Data

Two tasks:

- Divide (all of) space into chunks
- Map chunks of space to regions



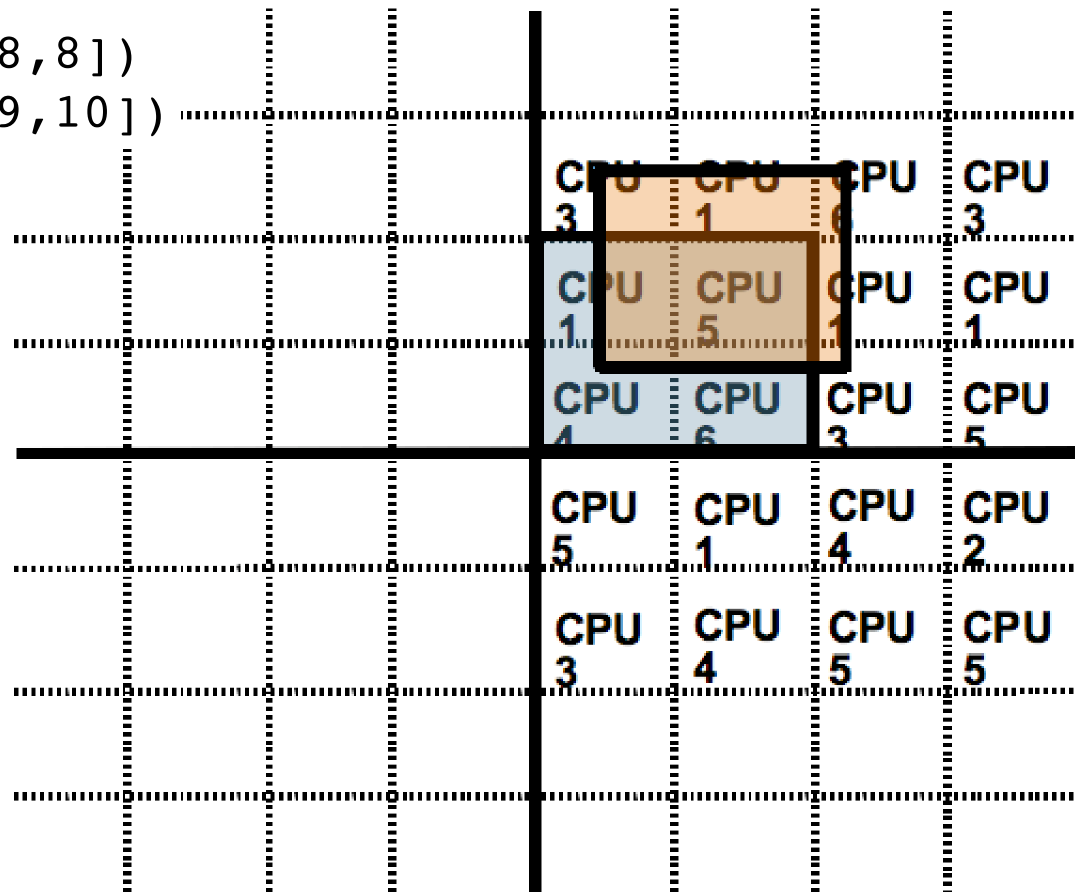
Division of space is recursive, proceeds one cut at a time.

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Distributions: Allocating Data

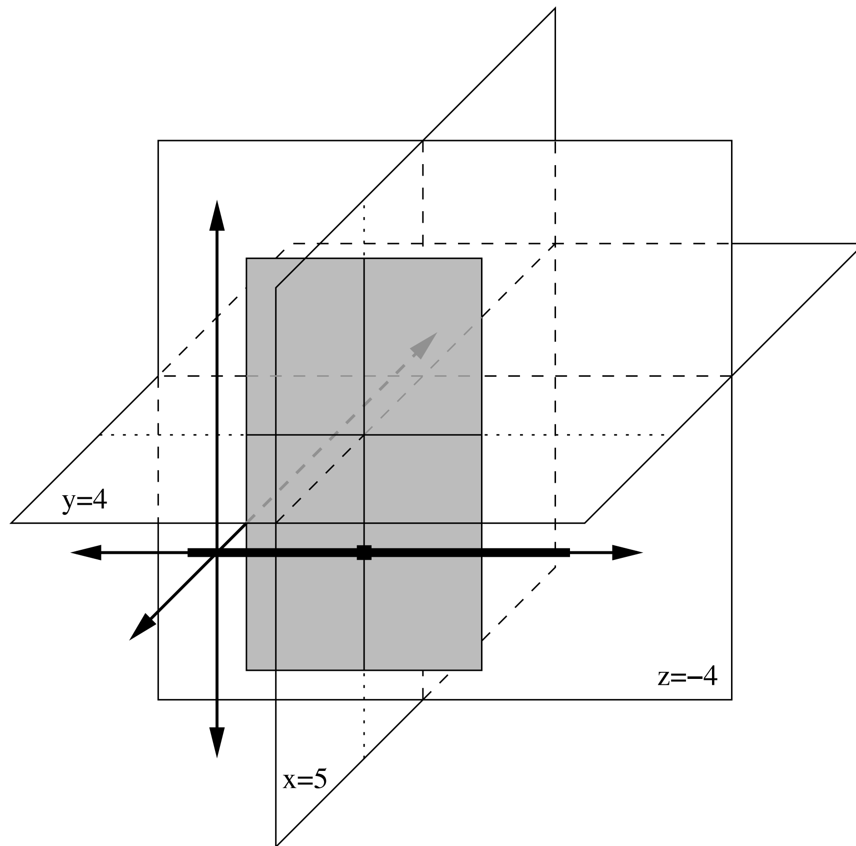
Allocation: d.array(l, u)

```
a := d.array([0,0], [8,8])
b := d.array([2,3], [9,10])
```



Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Distributions have infinite dimension

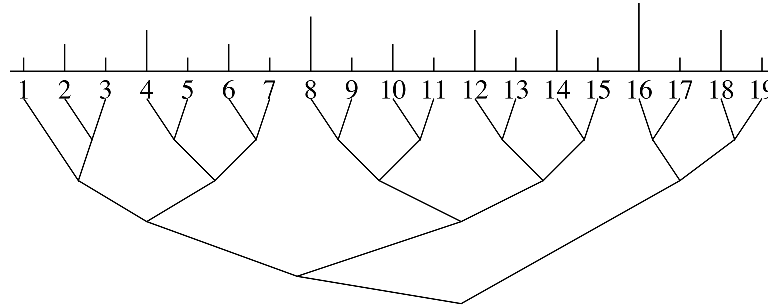


- Infinite-dimensional space
- Cut by sequence of planes
- Each orthgonal to an axis
- Each oriented

- Arrays and iteration spaces reside in lowest dimensions of this space

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

The Ruler: A Simple Subdivision



- Distribution gives rise to a binary tree
- Tree describes subdivision of particular subspace
- Tree need not be balanced
  - But a regular structure is a must
- Nodes of tree annotated with locality information
  - Schedule via locality-guided work stealing

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Current State of Play

- Language still in a state of flux
- Running simple programs in an interpreter
- Working on running Fortress on the Java™ Virtual Machine
- Prototype of distributions as a Java library
- Spinning up library effort
- Proving type soundness

Copyright © 2005 Sun Microsystems, Inc. All rights reserved.

Parallelism in Fortress

http://research.sun.com/projects/plrg

Carl Eastlund, Guy Steele, Jan-Willem Maessen, Yossi Lev, Eric Allen,
Joe Hallett, Sukyoung Ryu, Sam Tobin-Hochstadt, David Chase, João Dias