ZPL: An Easy-To-Use, High-Performance Programming Solution.

Angel de Vicente

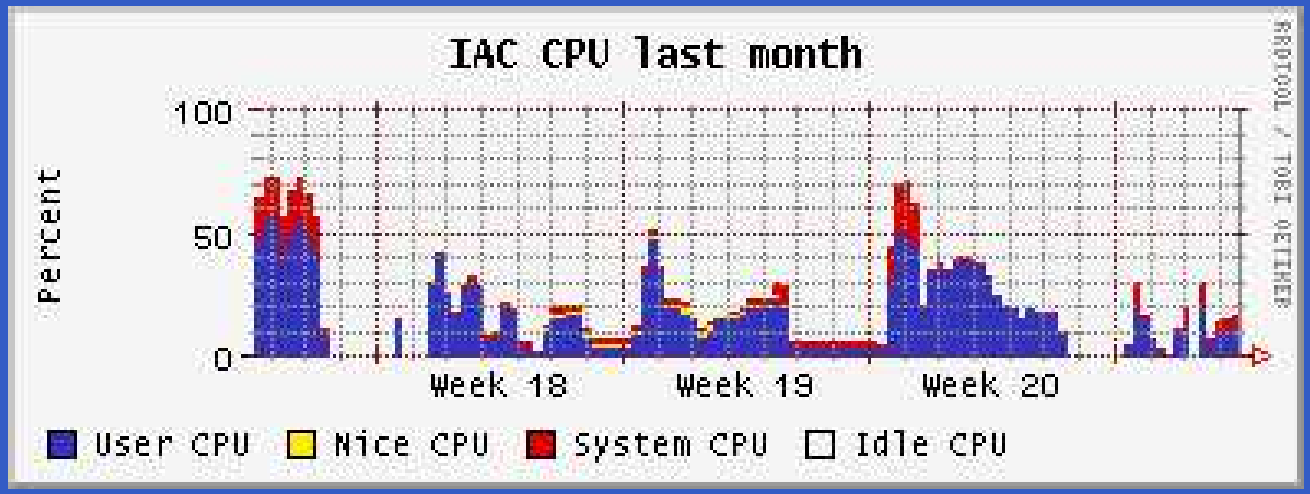
angelv@iac.es

Talk overview

- Why this talk?
- What is ZPL?
- Performance of ZPL?
- My first ZPL program
- Why you should try it
- Conclusions

Why this talk?

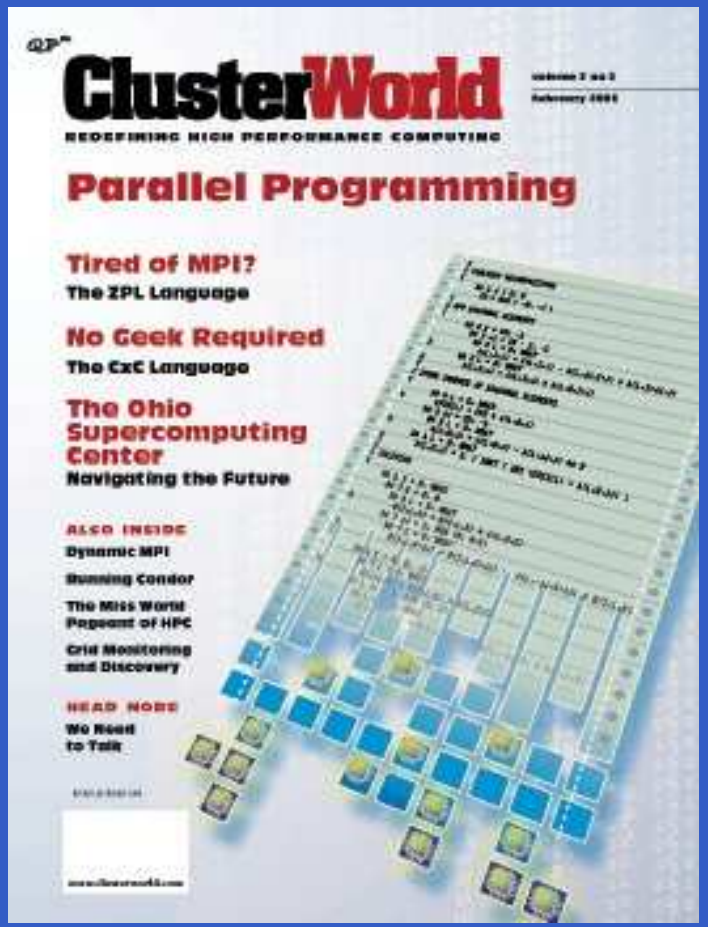
- Beoiac cluster underused (see http://beoiac)



- Parallel programming somewhat complex
- Several tools exist to make PP easier

Why this talk? (2)

- ZPL coming of age?



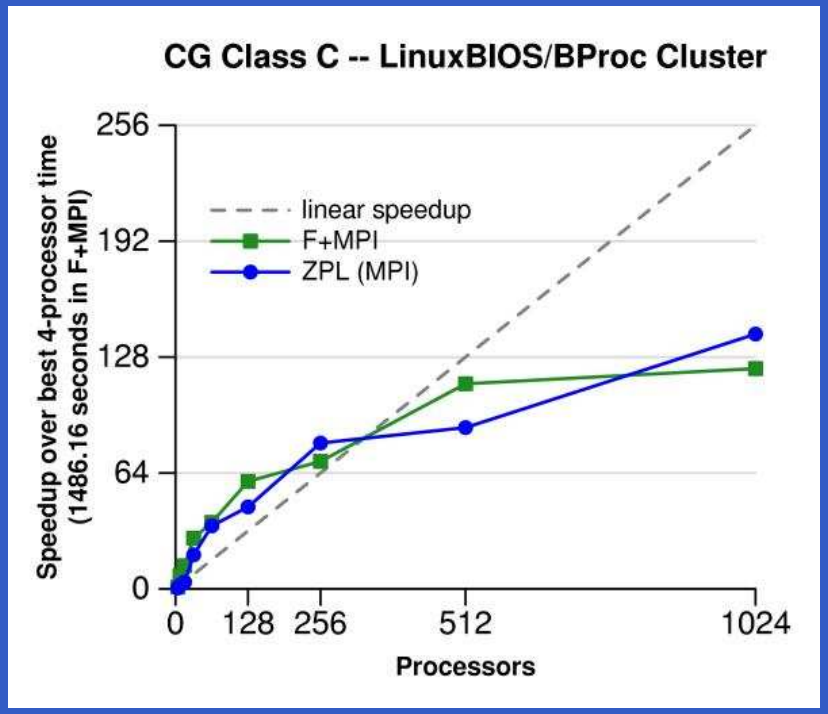
What is ZPL?

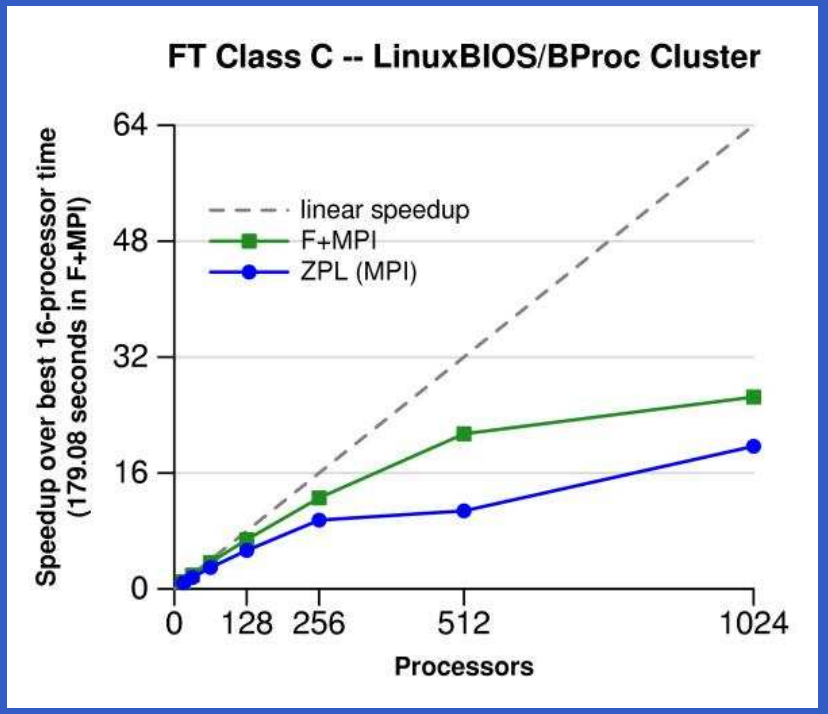
- ZPL is a parallel array programming language
- The false seduction of legacy code reuse
- ZPL starts from first principles
- Based on Modula-2 ...
  - ... to force you to start afresh
  - ... to make it readable and intuitive
- ZPL has its home at:

`www.cs.washington.edu/research/zpl/home/index.html`

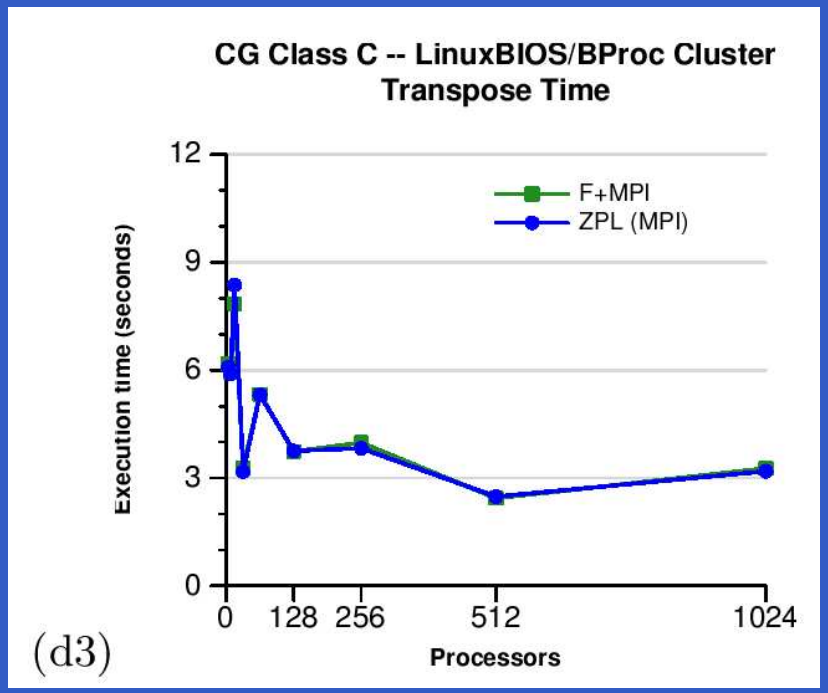
Performance of ZPL?

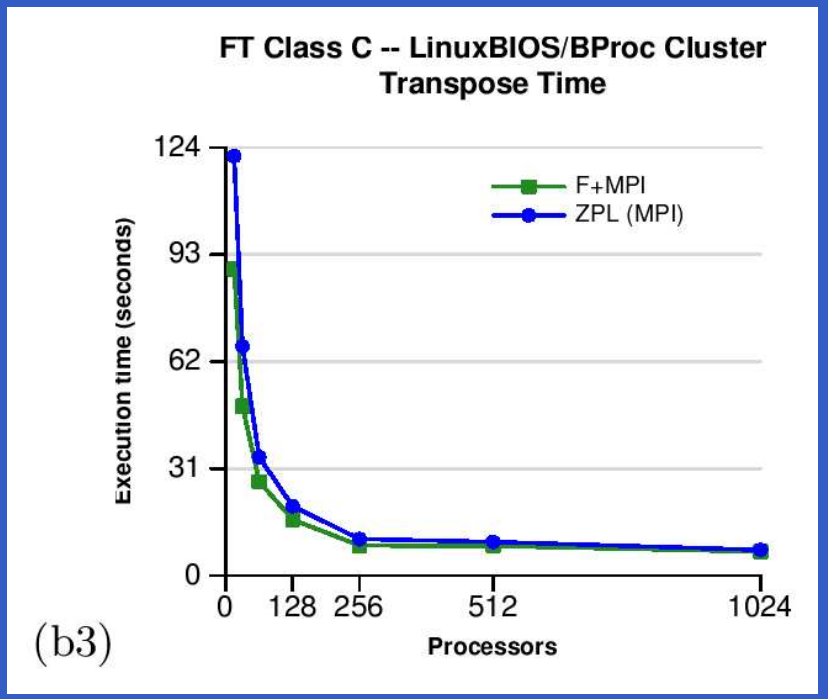
- NAS benchmarks (http://www.nas.nasa.gov/Software/NPB/)





Performance of ZPL? (2)





Details in:

''The Design and Implementation of a Parallel Array[...]''

by Deitz, S. J., et. al., 2003

Performance comparison in Beoiac

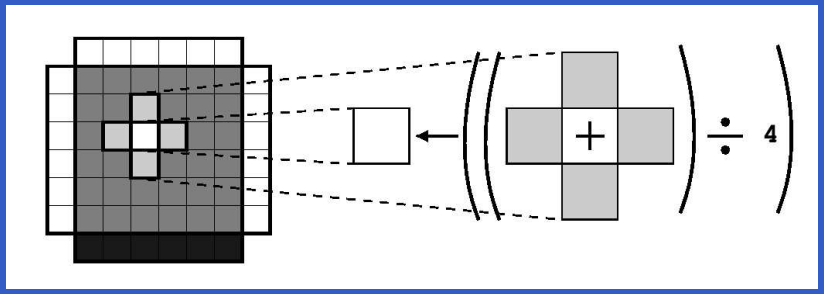
	Number of processors					
	2	4	8	16	32	64
FT (F+MPI)	(1)	(4)	(4)	117.83	180.94	182.91
FT (ZPL)	(2)	2018.98	517.10	271.41	300.30	311.78
CG (F+MPI)	1229.20	661.16	191.21	122.92	101.25	124.87
CG (ZPL)	(3)	(3)	(3)	127.17	124.25	118.46

(1) Compilation aborted
(2) not done, since (1)
(3) overflow
(4) Segmentation fault

Sample program in ZPL

- Jacobi iteration

  Given an array A, iteratively replace its elements with the average of their four nearest neighbours, until the largest change between two consecutive iterations is less than epsilon.



Jacobi iteration in ZPL

```
program jacobi;

config var    n       : integer = 512; -- problem size
              epsilon : float   = 0.0001;

region        R = [1..n, 1..n];       -- problem region

var           A, Temp: [R] float;
              err    : float;

direction     north = [-1,  0];
              east  = [ 0,  1];
              south = [ 1,  0];
              west  = [ 0, -1];
```

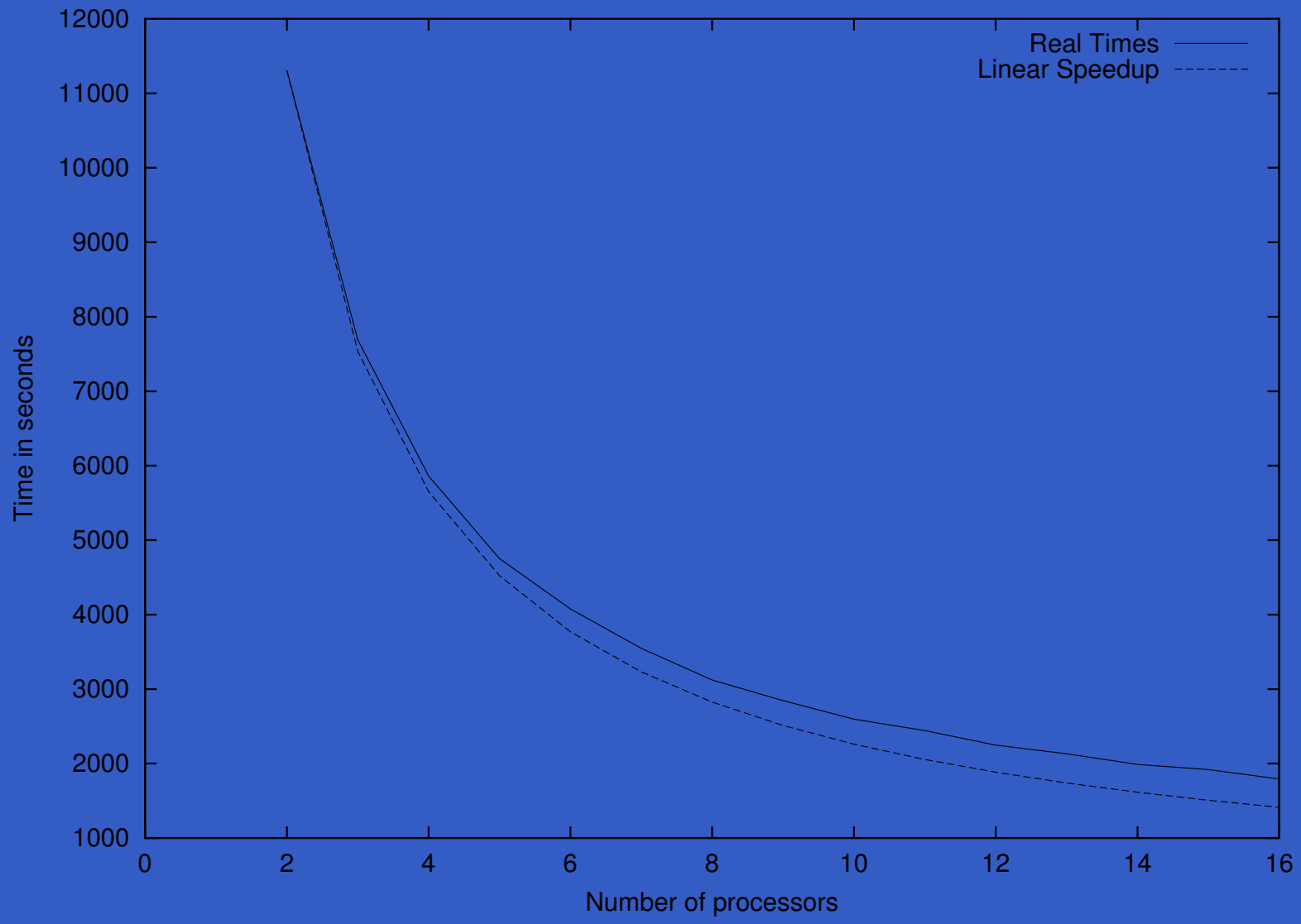
Jacobi iteration in ZPL (2)

```
procedure jacobi();
begin
  [R]           A := 0.0;  -- Initialization
  [north of R] A := 0.0;
  [east  of R] A := 0.0;
  [west  of R] A := 0.0;
  [south of R] A := 1.0;

  [R]           repeat
                  Temp := (A@north + A@east
                        + A@south + A@west) / 4.0;
                  err := max<< abs(A-Temp);
                  A := Temp;
                until err < epsilon;
end;
```

Demonstration of running ZPL

- Testing can be done in sequential mode.
  - Env. var's define the type of compilation.
  - C code can be saved if required.
- Execution can be done in parallel.

Speedup of Jacobi program?

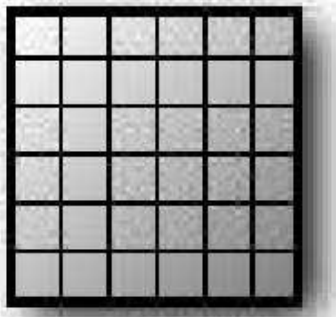


Preliminary ZPL Concepts

- Region, the key abstraction.

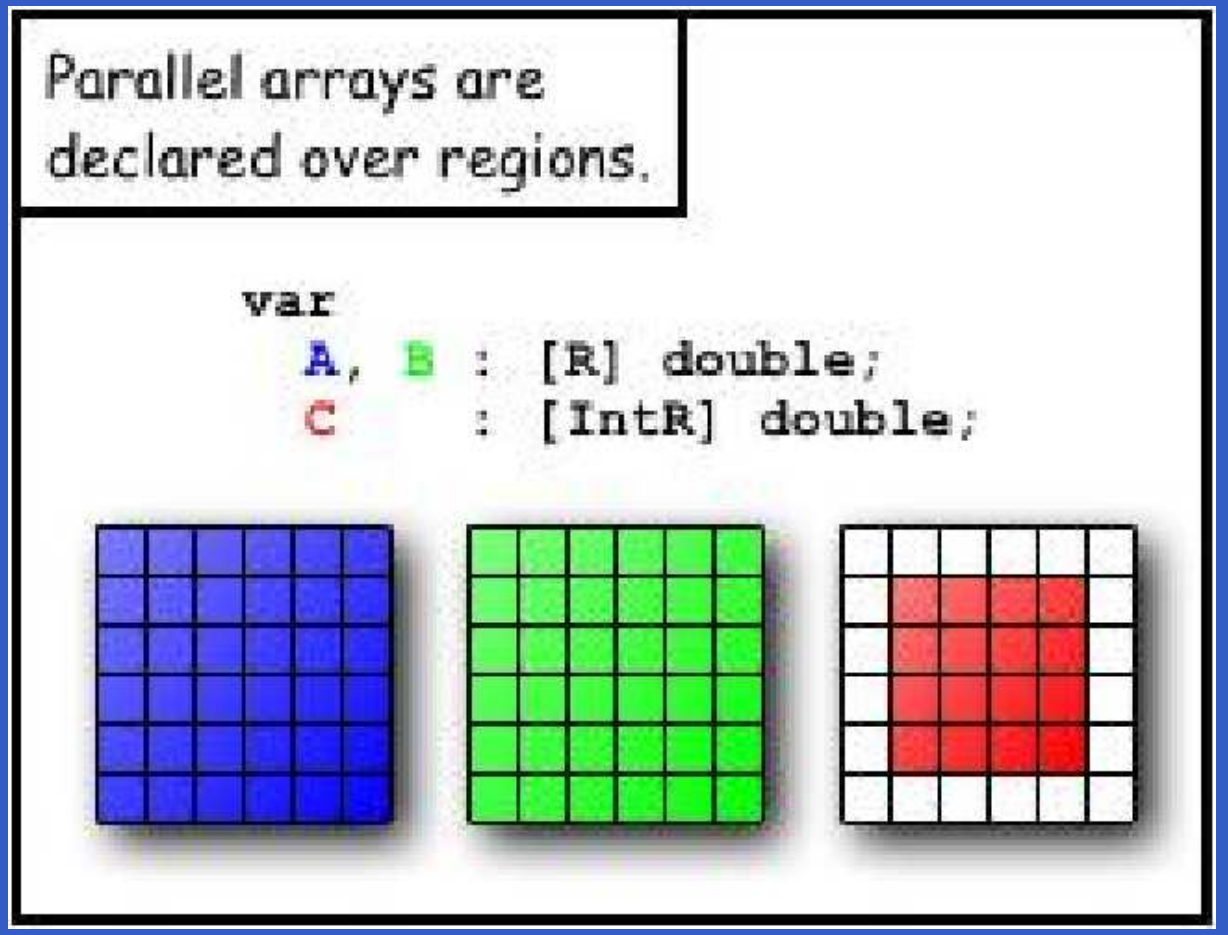
Regions are index sets. Region R is an n x n index set.

```
region
  R = [1..n,1..n];
```



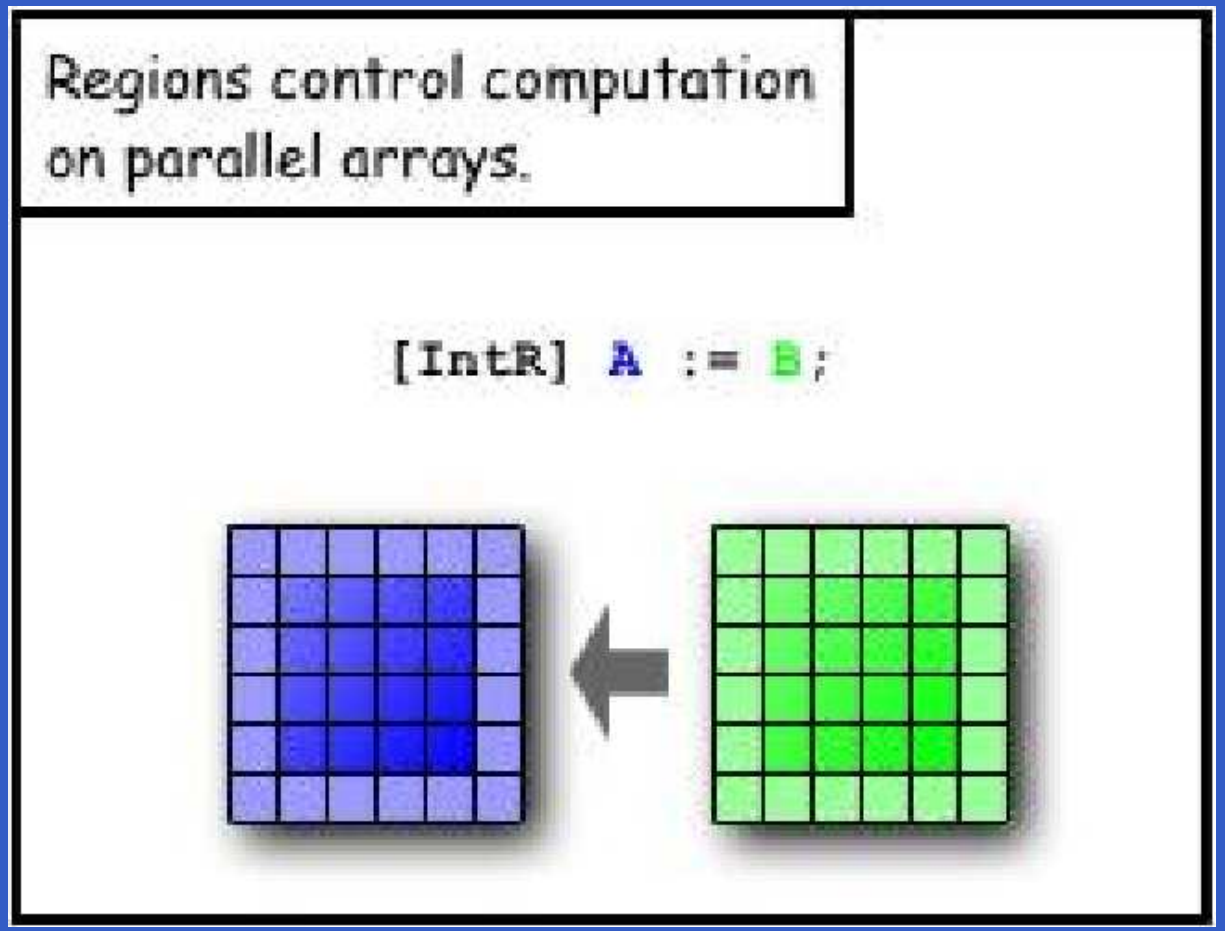
Preliminary ZPL Concepts (2)

- Arrays are created over regions.



Preliminary ZPL Concepts (3)

- Regions control computation.



Preliminary ZPL Concepts (4)

- Directions to manipulate regions.

Directions are offset vectors used to manipulate regions and array data.

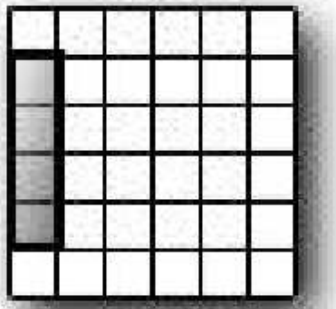
```
direction
  north = [-1,  0];
  east  = [ 0,  1];
  south = [ 1,  0];
  west  = [ 0, -1];

  nw = [-1, -1];
  ne = [-1,  1];
  sw = [ 1, -1];
  se = [ 1,  1];
```

Preliminary ZPL Concepts (5)

- Region operators: in, of, at, by

The "of" preposition creates a new region on the outside.

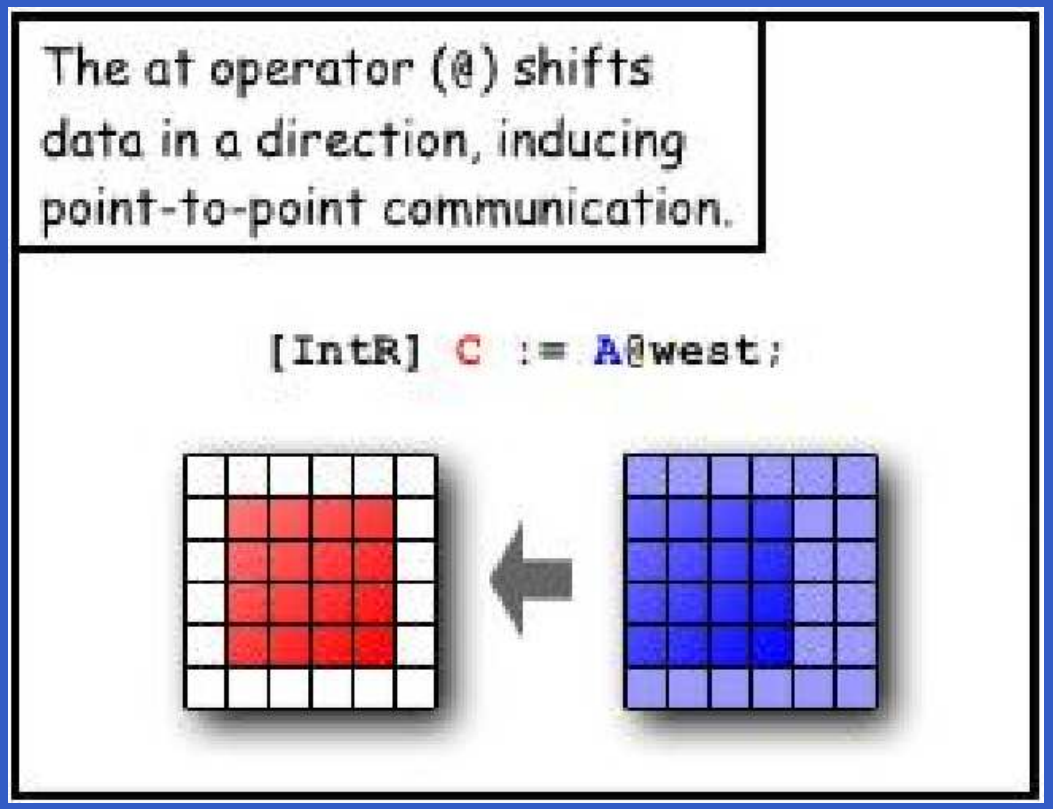
```
region
  SmallLeft =
    west of IntR;
```



Preliminary ZPL Concepts (6)

- Array operators:

  `@, @^, >>, <<, #, ||, wrap, reflect`

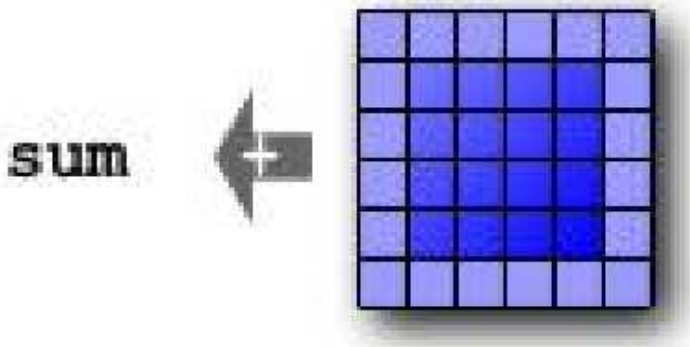


Preliminary ZPL Concepts (7)

- Reductions: Full reduction

The reduce operator (*op*<<) computes reductions using a combining operator and induces reduction communication..

```
[IntR] sum := +<< A;
```

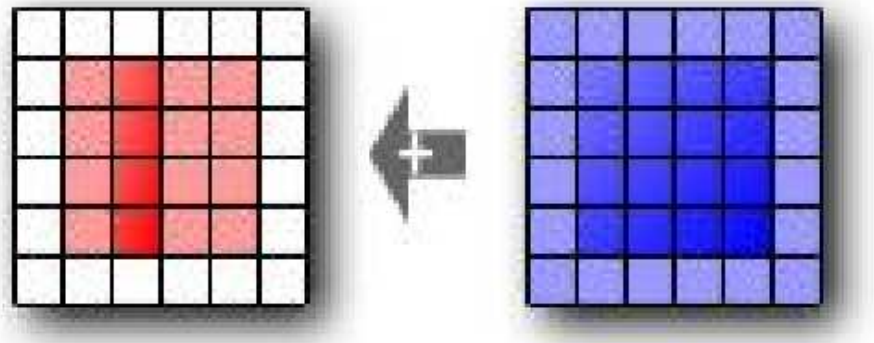


Preliminary ZPL Concepts (7)

- Reductions: Partial reduction

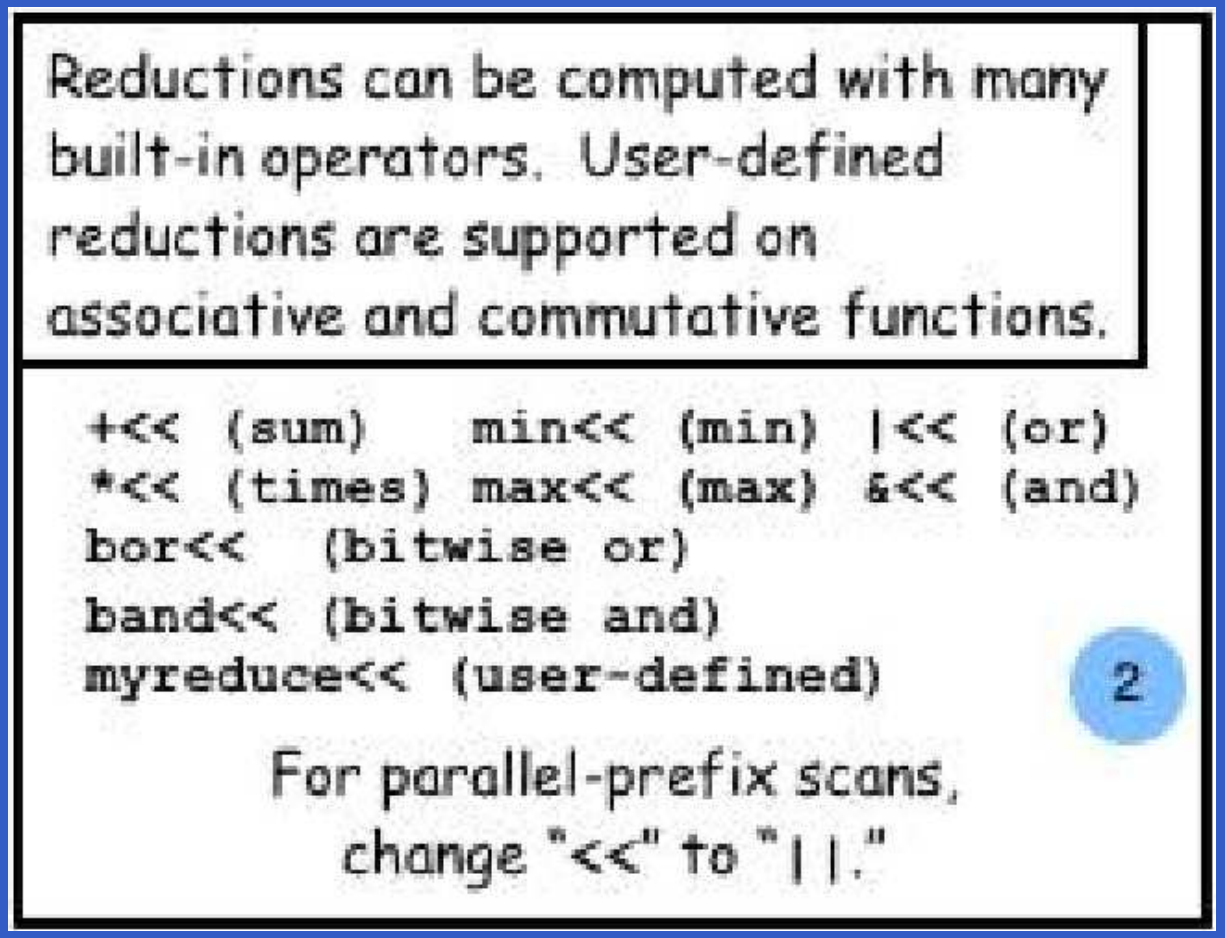
A partial reduction collapses dimensions of an array.

```
[2..n-1,i] C := +<<[IntR] A;
```



Preliminary ZPL Concepts (8)

- Reductions can be user-defined



No time to learn a new language?

- Learning MPI will probably take you longer
- Writing MPI will take you longer
- Debugging MPI will take you longer
- Performance competitive
- Compare codes ...

Conclusions

- ZPL is a convenient high-level programming medium for supercomputers.
- It can make your life easier.
- Installed in the Beoiac.
- Sufficient documentation

  ‘‘A Programmer’s Guide to ZPL’’
  by Lawrence Snyder
- Anyone wanting to try it, contact me.

That's all folks ...

- Thanks!