ZPL

• No explicit threads
  ◦ no operation like `omp_get_thread_num` or `Ti.thisProc`

• Only way to use an array is in a parallel operation

```plaintext
a : [0..N] integer;
sum : integer;

[0..N] a := a + 1;
[0..N] sum := +<<a;
```
Sortof like...

... OpenMP, in that the programmer declares places for automatic parallelism

ZPL “declarations” are much more fine grained, with more kinds of operators

... Titanium, in that communication is implicit through shared objects

ZPL automatically localizes data and has a different way of describing costs

... APL, in that good programs need to use the right operators

ZPL has fewer operators targeted just as parallelism
The following slides are based on the ZPL “comic”
Regions

region
   IntR = [2..n-1, 2..n-1];
Regions

region
Left = [1..n,1];
Directions

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

  nw  = [-1,-1];
  ne  = [-1, 1];
  sw  = [ 1,-1];
  se  = [ 1, 1];
Region Operators

region
Left = west in R;
Region Operators

region
SmallLeft = west of IntR;
Region Operators

region
  \text{IntRLeft} = \text{IntR} \text{ at west};
Region Operators

direction
step = [1,2];

region
SR = R by step;
Declaring Arrays

var
    A, B : [R] double;
    C : [IntR] double;
Regions Control Statements

[IntR] A := B;
Regions Control Statements

[IntR] C := A + B;
Regions Control Statements

[IntR] C := A@west;
Reduction

\[ \text{IntR} \quad \text{sum} := +\langle\rangle \ A; \]
Partial Reduction

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

[R] $B := A#[\text{Index2}, \text{Index1}]$
ZPL’s performance model specifications for worst-case behavior; the actual performance is influenced by $n$, $P$, process arrangement, and compiler optimizations, in addition to the physical features of the computer.

<table>
<thead>
<tr>
<th>Syntactic Cue</th>
<th>Example</th>
<th>Parallelism $(P)$</th>
<th>Communication Cost</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[R] array ops</td>
<td>[R] ... A+B ...</td>
<td>full; work/$P$</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>@ array transl.</td>
<td>... A@east ...</td>
<td>-</td>
<td>1 point-to-point</td>
<td>xmit “surface” only</td>
</tr>
<tr>
<td>&lt;&lt; reduction</td>
<td>... +&lt;&lt;A ...</td>
<td>work/$P + \log P$</td>
<td>2 log $P$ point-to-point</td>
<td>fan-in/out trees</td>
</tr>
<tr>
<td>&lt;&lt; partial red</td>
<td>... +&lt;&lt;[ ] A ...</td>
<td>work/$P + \log P$</td>
<td>log $P$ point-to-point</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>scan</td>
<td>... +</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt; flood</td>
<td>... &gt;&gt;[ ] A...</td>
<td>-</td>
<td>multicast in dimension</td>
<td>data not replicated</td>
</tr>
<tr>
<td># remap</td>
<td>... A# [I1, I2] ...</td>
<td>-</td>
<td>2 all-to-all, potentially</td>
<td>general data reorg.</td>
</tr>
</tbody>
</table>