Changes to Parallel JavaScript (River Trail)
Map

• myArray.map(elementalFunction)
• myArray.map(depth, elementalFunction) // for an n-dimensional array
  – elementalFunction (element, index, source) // similar to Array.map
  – If depth is provided index is a vector holding the depth indices
  – Otherwise index is a scalar into top level
• Alternative was to add a new ParallelMatrix type for the N-dimensional case
• ParallelArray is agnostic about the value of |this| in elementalFunction
  – Use of ES6 function syntax => expected and over riding |this| would complicate semantics
Examples of map

- `paArray.map(
    function(element){return element+1;});
  // increments each element
- `paArray.map(2,
    function(element){
        return element+1;});
  // increments each element in 2D ParallelArray
- `myArray.map(2,
    function(element, [i, j], array){
        return array[i][j] + 1;});
  // increments each element in 2D ParallelArray, uses arg destructuring
- Alternative signature if rest parameters would be allowed in the middle of function parameter lists.
  - `map(d, function (e,i,j,k,v) {...})` for ND if rest parameters will be allowed in the middle of function parameter lists
Shape

• Mixing 1D and 2D operations requires an understanding of shape
• Shape is a dynamically property determined at construction
• Shape describes the maximum regular array
• Leaf elements will never consist of ParallelArrays all of which have the same length
Identity

- Accesses to non leaf elements of a ParallelArrays will return a freshly minted ParallelArray
- Reference semantics for \(===\) remains consistent
- \(pa[2] === pa[2]\) true only for 1D ParallelArrays
- \(pa[2] === pa[2]\) always false when shape is \(> 1\)