HW3 Report – Array Language

許峻源
111062649

The Benefit Of Array Language

Higher Level of Abstraction:

Array languages simplify programming by providing a more concise and convenient way to write code. This is achieved by abstracting away complex control flows, making code easier to write and understand.

Natural Source of Data Parallelism:

These languages inherently support concurrent manipulation of multiple array elements. This parallelism is a significant advantage in processing large datasets efficiently, as it allows for simultaneous operations on different parts of an array.

Various Array Languages

1. **APL**:
   Known for its approach of applying operators to all elements of their array operands without a method for accessing portions of an array.

2. **Fortran 90**:
   Introduces array *slices* for concise specification of indices and provides additional operators for manipulating arrays; however, they can be error prone.

3. **C***:
   Designed for the Connection machine, it simplifies array indexing with indices relative to each array element, which is less error prone than *slice.*

4. **ZPL**:

   Raises the level of abstraction by using *regions(South, North, West, East)* to represent index sets and employs the @ operator for indexing.

5. **Chapel**:

   Developed by Cray, offering a form of relative indexing adaptable to various formats, such as “A[R+north]” or “A(ij+south)” etc.

6. **FIDIL**:

   Supports irregular grids with manipulable index sets known as *domains*, Which can be manipulated through union, intersection, and difference Operators.

Array Operator

All array languages enable elementwise operations, extending scalar operators to array operands. They often include reduction operators for summarizing data, converting multiple array values into a single scalar. Some languages enhance this by allowing reductions on specific array dimensions. Additionally, parallel prefix or scan operators extend reductions, generating an array of partial values, not just a scalar. This operator is potent for parallelizing computations that might appear to require sequential iteration.

The benefits from the language support are not only easier reading and writing, but also more customizable, and compiler support for nontrivial opportunities for optimization.

My Reflection

Although this is not a really lengthy paper, it has provided me a lot of concepts about the array language from this paper, such as the various types of array languages, and the distinct main functions each focuses on. With the support of these array languages, it will be easier to write readable code. Most people can comprehend the code without putting lots of effort to track the loops. After becoming familiar with these languages, I can incorporate them in my code, which can make my work more readable, parallelizable, and efficient.