Programming Model and Architecture for Real Time Streaming

Jack B. Dennis
MIT Computer Science and Artificial Intelligence Laboratory
dennis@csail.mit.edu

February 20, 2016

The Fresh Breeze project[1, 2] is developing a programming model and system architecture that supports several forms of parallel computation without the burden of operating system and runtime software. The principal forms of parallelism supported are producer/consumer parallelism for streaming computations and data parallel processing for classical HPC applications. User programs are written in funJava, a functional variant of Java; the Fresh Breeze compiler[4] converts funJava programs into codelets for execution by a simulated Fresh Breeze multi-core processor.

An example will illustrate the programming style used for writing streaming applications in funJava. We consider the simple stream computation illustrated in Figure 1 consisting of two modules that are sources of data streams, a merge unit that combines the two streams into a single stream processed by a filter module, and an analysis module that summarizes properties of stream data elements. To keep things simple, the stream elements are Floats, although funJava permits any element type to be used. Here is how the top level (main) program may be written:

```
static void main (String[] args) {
    Stream<Float> mergedData =
        Stream.merge (sourceOne(), sourceTwo());
    analyze (filter(mergedData);
}
```

The four stream processing methods `sourceOne`, `sourceTwo`, `filter`, and `analyze` are written using operations of a special `Stream` class having methods with type signatures as follows:

```
class Stream {
    Stream streamCreate ()
    Stream Append (Float element)
    Float First ()
    Stream Rest ()
}
```

The methods of the Stream class are implemented directly as Fresh Breeze machine instructions. Using these methods, the sourceOne function (which generates a stream containing a specified number of consecutive integers) may be written as:

```
Stream sourceOne (int count) {
    Stream sourceStream = new Stream();
    for (int i = 0; i < count; i++) {
        Float x = (Float) i;
        sourceStream = sourceStream.Append(x);
    }
    return sourceStream;
}
```

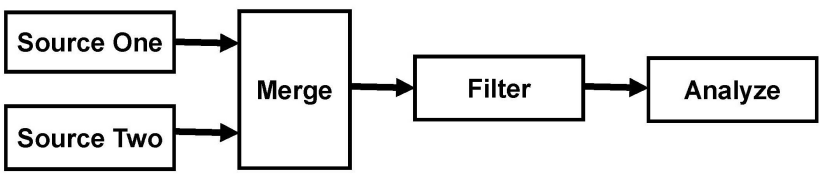

Figure 1: Diagram of a simple stream processing computation.

The filter function (which computes a weighted sum of three elements of its input data) may be written as:

```
Stream filter ( Stream strm0 ) {
    Float x0 = strm0.First();
    Stream strm1 = data.Rest();
    Float  x1 = strm1.First();
    Stream strm2 = data.Rest();
    Float  x2 = strm2.First();

    Stream newStrm = averagePairs (strm1);
    newStrm.streamAppend(0.25 + x0 + 0.5 + x1 + 0.25 + x2);

    return newStrm;
}
```

1 Implementation

Performing computations on data streams that include real time interactions with devices or observers requires a computing platform capable of managing dynamic changes in resource demand by several parts of application software. The Fresh Breeze programming model and system architecture have been developed to support execution of such computations with high performance and energy efficiency. The Fresh Breeze system architecture is designed to provide very efficient automatic management of memory and tasks using hardware mechanisms, For memory this is achieved by representing all data objects by trees of fixed-size chunks of memory. The trees have a fan-out of 16, so a vector of 256 data elements is represented by a tree of chunks with depth = 1: sixteen chunks, each holding 16 data elements. A data stream is naturally represented by a chain of chunks, each memory chunk holding several stream data elements. Data elements are removed from the head of the chain and added at the tail. Synchronization for the case that a remove is attempted from an empty stream is implemented using futures.

The Fresh Breeze multi-core architecture uses an instruction set that directly supports operations on trees of chunks and the scheduling of tasks for codelet execution. There is no operating system or runtime software to add to overhead and increase energy consumption.

Our current simulation model of a multi-core Fresh Breeze processor has demonstrated linear speedup for up to at least 256 processing cores for matrix multiplication. We expect to show that a Fresh Breeze system will perform stream processing for many data streams concurrently with conventional HPC, achieving stream processing rates approaching gigabytes per second. We are interested in exploring challenging applications with funJava and the Fresh Breeze system architecture.

References

[1] J. B. Dennis. A parallel program execution model supporting modular software construction. In *Massively Parallel Programming Models*, pages 50–60. IEEE, 1997.

[2] J. B. Dennis. Fresh Breeze: a multiprocessor chip architecture guided by modular programming principles. *ACM SIGARCH Computer Architecture News*, 31(1):7–15, 2003.

[3] J. B. Dennis, G. R. Gao, X. X. Meng, B. Lucas, and J. Slucom, "The Fresh Breeze program execution model," in *Parallel Computing, Ghent, Belgium*, August 2011.

[4] J. B. Dennis, "Compiling Fresh Breeze codelets," in *Proceedings of Programming Models and Applications on Multicores and Manycores*, PMAM'14, (New York, NY, USA), pp. 51:51–51:60, ACM, 2014.

[5] , J. B. Dennis, "Stream data types for signal processing," in *Advances in Dataflow Architecture and Multithreading*, Eds. J.-L. Gaudiot and L. Bic, IEEE Computer Society Press, 1995.