Introduction
Distributed stream processing requires a combination of technologies to monitor workloads and map them onto the local resources of worker machines. Here we highlight three sub-projects that address distinct aspects of this problem:

- Compile JIT compiler to create dynamic dataflow graphs for available parallel architectures (Accelerate)
- Communicate: Sending irregular data efficiently between nodes (Compact Normal Form)
- Monitor: Profiling native-code programs based on binary self-modification and cross-modification (UbiProf)

Accelerate
In a distributed execution plan, a subgraph of a stream dataflow graph must map onto the hardware of a worker node and achieve throughput. Our approach is a DSL JIT compiler called Accelerate.

- Accelerate [6] takes a graph of data transformations and generates CUDA or LLVM code to run on CPU or GPU.
- Accelerate can launch concurrent GPU kernels on each new input (stream element) that arrives.
- Accelerate is a (partially) formally verified compiler [4].

Currently, we apply Accelerate to streaming problems in the video domain:

Via it’s LLVM backend, Accelerate can run data-processing algorithms on the CPU too.

This work is supported by NSF XPS award #1337242, "XPS: D2S: Adaptive Stream-Processing Compilers for a Messy World"