# Seamless Dynamic Runtime Reconfiguration in a Software-Defined Radio

Michael L Dickens, J Nicholas Laneman, and Brian P Dunn



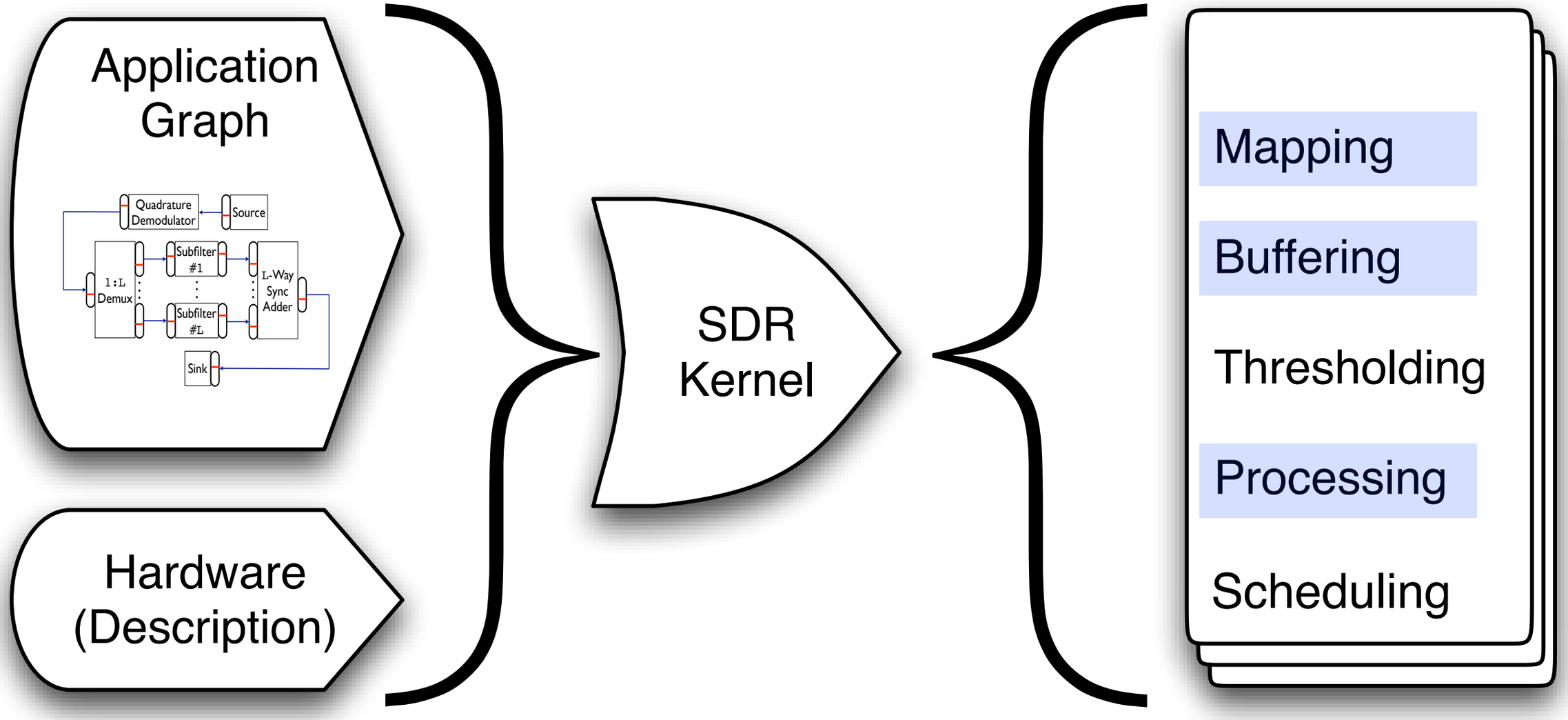



WINNF'11'Europe 2011-Jun-22/24

# Overview

- Background on relevant SDR
- Problem formulation
- 3 issues and our solutions
- Example: NBFM decoding / CPU load
- Conclusions

# SDR Kernel Role



# Relevant Work

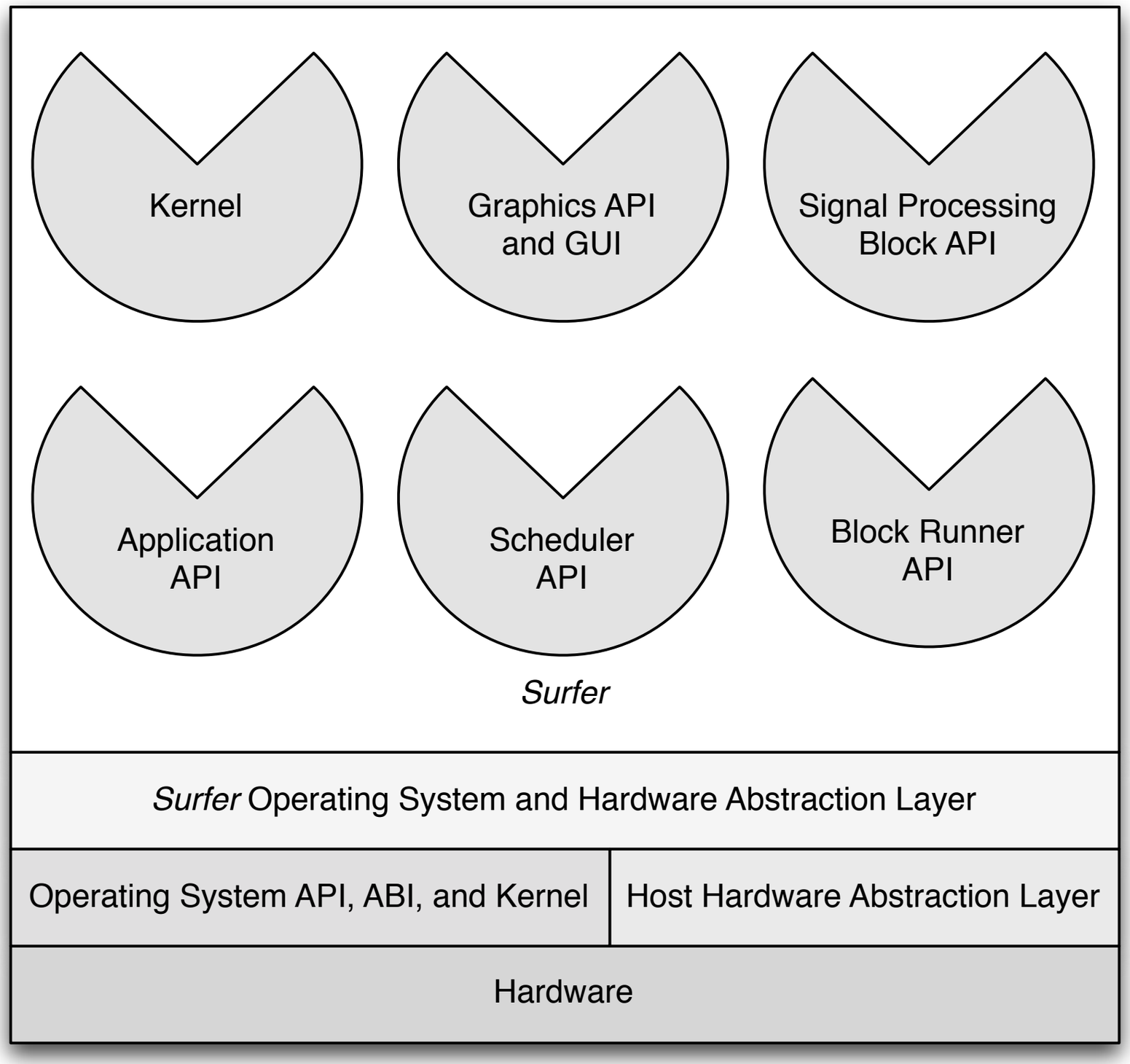
## GNU Radio

- Single shared GPP (C2D, Cell, PPC, Arm)
- Ettus E series: could allow heterogeneous processing
- Continuing to work towards GPU using NVIDIA CUDA
- Each signal processing block is processor specific

## SCA / CORBA

- Any processor that can be interfaced into CORBA
- Heterogeneous processing via a-priori scheduling
- Full availability of CPU, GPU, DSP, FPGA, …
- Each signal processing block is processor specific

# Our Work: *Surfer* '10




- Shared and/or dedicated processors
- Pool of Runner Threads
- Separated block overhead from processing
- Block threshold determined globally
- Overhead load distributed across all threads
- Runtime statistics collection

# Problem Statement

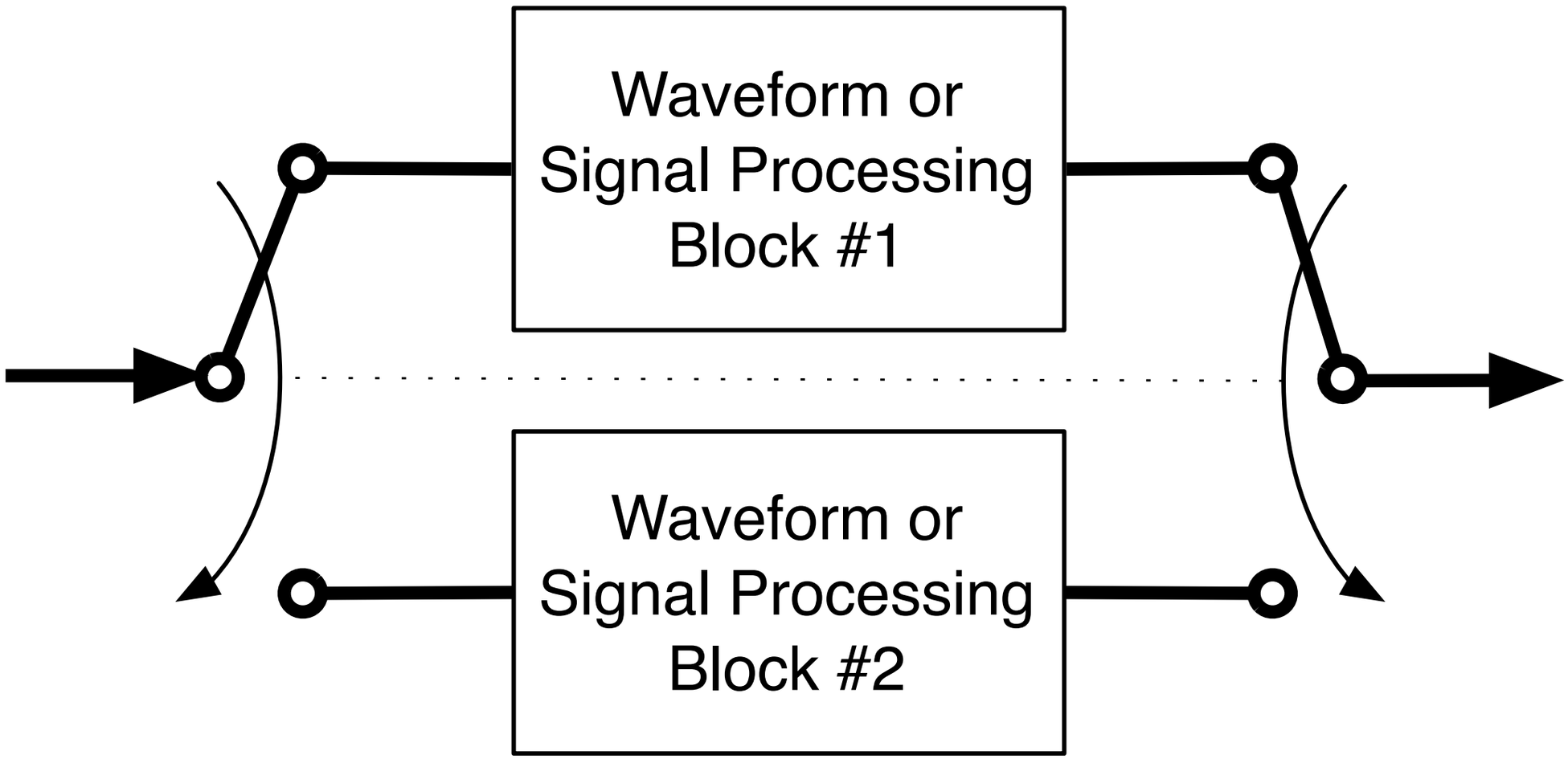
To allow waveform reconfiguration at the block level, during runtime, without interrupting data processing

To allow signal processing to be switched between processing devices while maintaining state

➡ Requires a switch … somewhere …

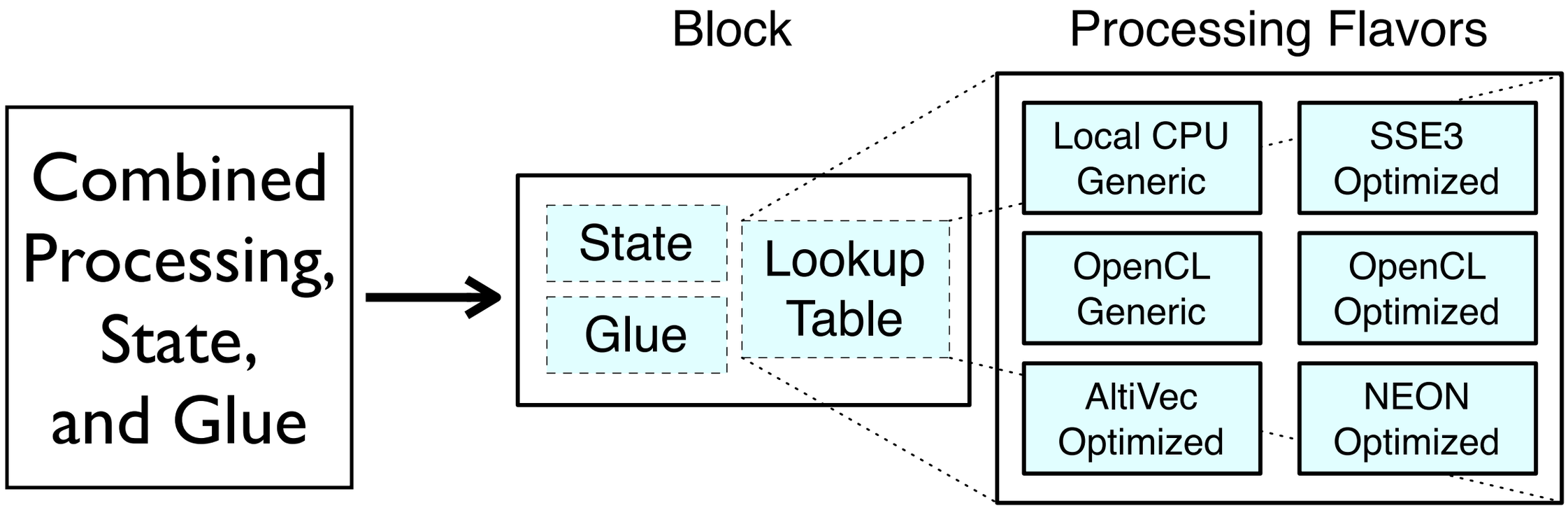
# Primary Issues

## Where and how to switch?



# Our Solution

## Split blocks




- Processing is cleanly separated from state
- Processing flavor can be optimized for any device
- Processing can take place on any device
- User can modify lookup table with new processing flavors

# Another Issue

## How to keep state?

- Must be easily copyable; require contiguous memory
- Must be fully interpretable by any processor
- Must provide alignment for each variable in structure
- C++ API should closely match that for scalars, std::vector, std::string
- Memory must be resizable, e.g., to accommodate vectors and strings

# Our Solution

Create processor-agnostic *dynamic structure*

## State

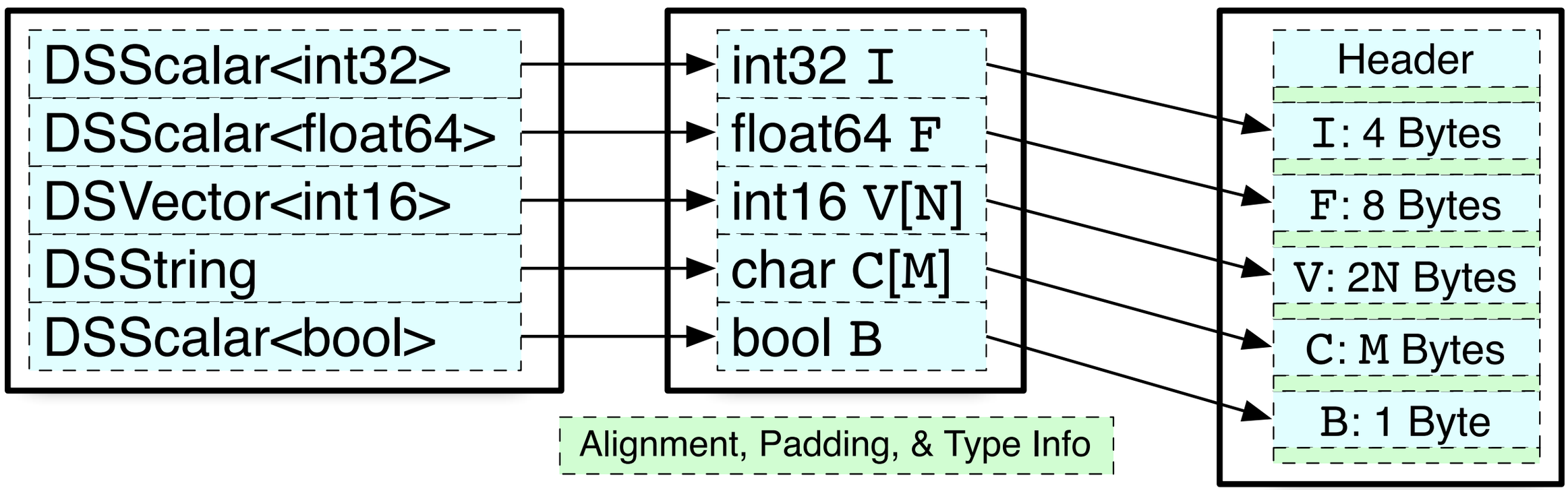
Static
Handles

## Variables

Dynamic
Pointers

## Allocation

Contiguous
Memory



# Final Issue

How to control flavor selection?

- ➡ Currently uses a *Scheduler*
- ➡ Need a way to use
  - A-priori collected statistics
  - Runtime collected statistics
- ➡ Need a way to control
  - Computation flavor selection
  - Internal *Surfer* parameters

# Our Solution

Create *supervisor* to handle monitoring system

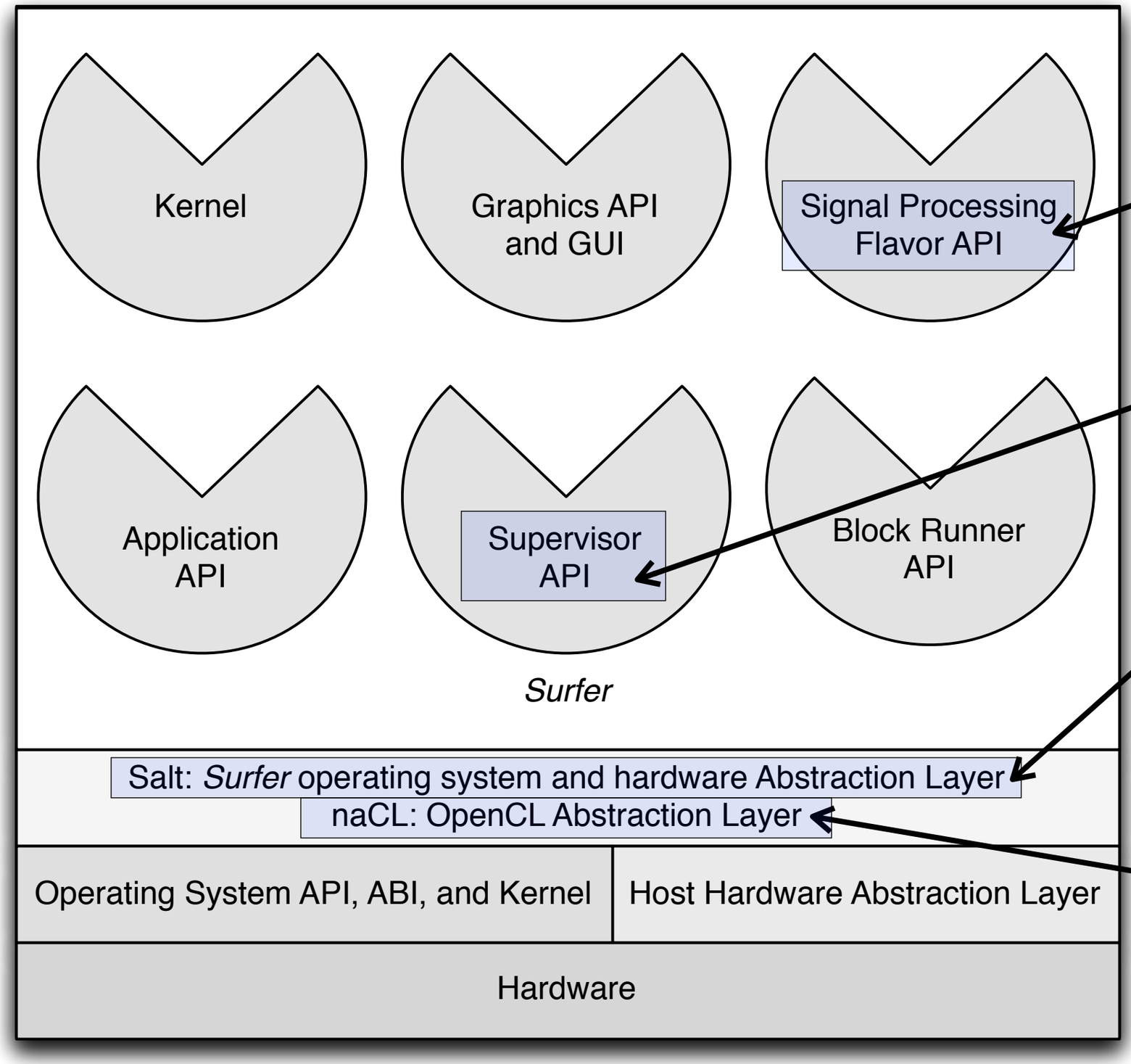
➡ Plugins include statistics collection for

- Host CPU load (per core)
- Computation throughput (per block)
- Data transfer throughput and latency
- *Surfer* load (per thread)
- Input / output data latency (per block)
- System overhead

➡ Can control

- Computation flavor selection (per block)
- Threshold settings (per block)
- Any other *Surfer* parameters

# *Surfer* '11 Changes



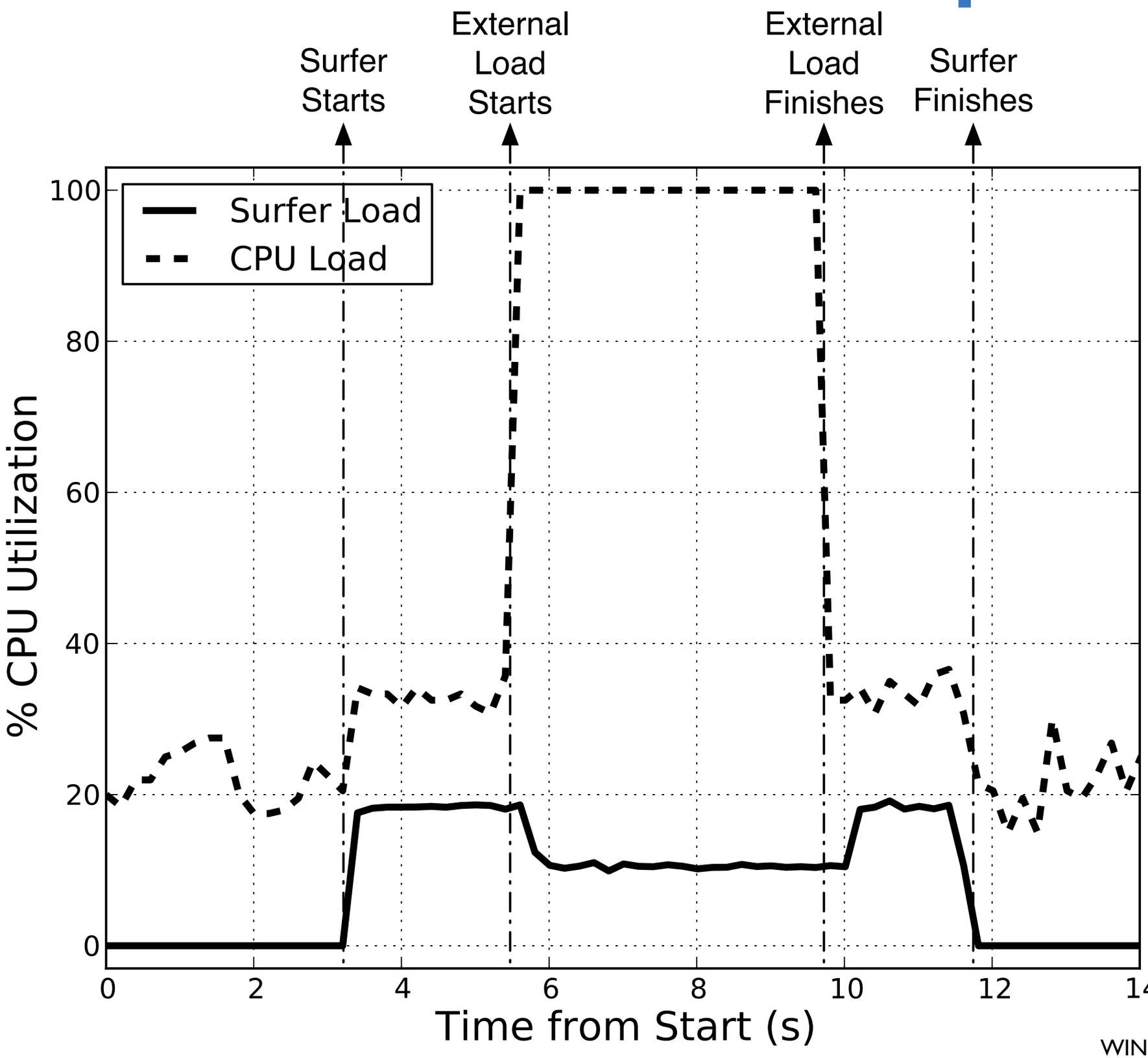

- Signal Processing *Block* becomes *Flavor*
- *Scheduler* becomes *Supervisor*
- New C++ namespace for OS and hardware abstraction: *Salt*
- New C++ namespace for OpenCL abstraction: *naCL*

# Example

1. Start CPU load collection
2. Start *Surfer*
   - Supervisor set to switch block flavors when host CPU load reaches 60% for 3 consecutive samples (0.3 seconds)
   - Decoding NBFM, ~9 seconds of data
3. Wait 1.5 seconds
4. Start CPU load hog, runs for ~4 seconds

➡ Expect to see *Surfer* CPU load decrease when CPU load hog is executing

# CPU Load Graph



# Conclusions

- Enabled seamless dynamic runtime reconfiguration
  - Split block implementations
  - Portable state construct
  - Supervisor to collect statistics and modify system runtime behavior