Dennis J. Nicklaus
For the Fermilab Front-End Group
ICALEPCS 2011

AN ERLANG-BASED FRONT END FRAMEWORK FOR ACCELERATOR CONTROLS

Duties of Front End Computers

- Read and set hardware, field-bus devices
- Communicate with rest of controls system
- Respond to timing system

Front End Framework Software

- Provide for different field bus device drivers
- Map between hardware and database
- Support standard communication protocols
- Run other algorithms (e.g. PID loops)
- Manage multiple connections
- Alarm limit checking, alarm posting

Fermilab’s Old Framework

- Functional, but not pretty (“C++ written in C”)
- Locked to VxWorks
- Core is mostly reliable, but
- Shared memory model means one rogue process can take down the whole front-end.
- Driver developers responsible for many mundane tasks: argument checking, alignment issues, …

The New Fermilab Framework

- Erlang and Linux Based
- Totally re-implemented
- Functional programming

Functional Languages

()()()((())))(()))

((((()))))(((())))

() () () () () () (()) () () () () () ((())) (()) ((()))

[[[]]] {{{}}}} [[[[]]]]

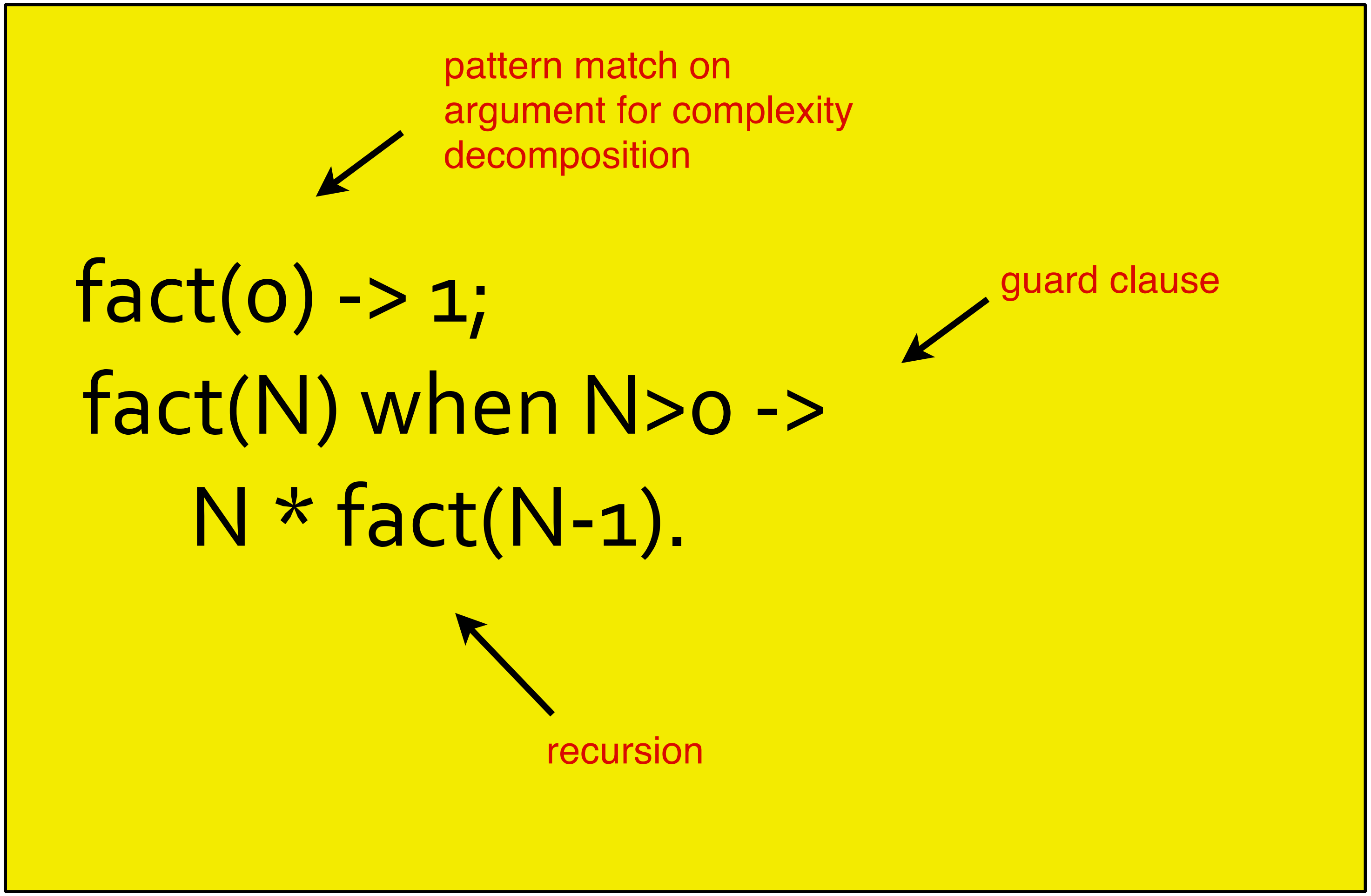
{ { [[[ ]]], [[]], [ ] ,[ ] } ,{ [[,[[ ]]],{ [ ] }] }

More Than Just Parentheses!

Some Erlang Syntax

```
A ! {b(C,D)}.
 f1(a(b(c())) )
{[1,[2,[3,4,[5,6]]]],[[[]]]}
[] -> CPid ! {e, 100};    [{_, V}] ->  CPid ! {ok, 1}
 <<0,0>> <<0:16, DI:32, Val:16>>,
{M,S,U} = now(),
f([]) -> [];f([{H1,H2}|T]) -> {<<H1:32>>,f(T)}.
```

Basic Erlang Functions



pattern match on argument for complexity decomposition

guard clause

```
fact(0) -> 1;
fact(N) when N>0 ->
     N * fact(N-1).
```

recursion

Erlang Brings:

- Proven reliability
- Real-time performance under Linux
- Functional style encourages testing, proof of accuracy
- High availability, distributed system
- High-level language concepts

Erlang Higher Level Concepts

- Lists (of course) and tuples
  - [1,2,3] and {1,2,3}
- Pattern Matching
  - {A,B,C} = now().
- Simplified Concurrency
- Message Passing
  - Pid ! Message
  - receive message1 -> f1(); message2 -> f2() end.
- Records (structured lists), list manipulation

Erlang: What we like

- Processes are cheap
- Interactive shell helps productivity
- Predefined process behaviours
  - supervisor
  - gen_server
- List manipulation for lots of connections to lots of devices.
- Functional encourages modular
- Testing philosophy
- No pointers

Erlang Helps With:

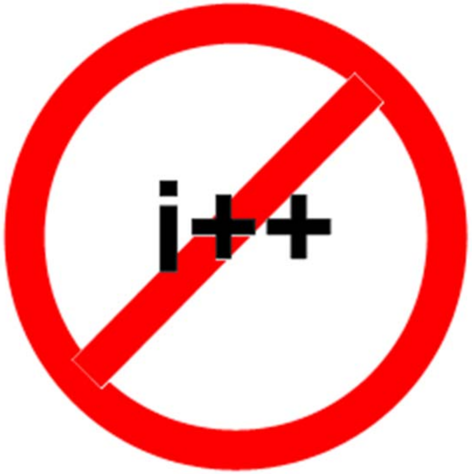
- Process monitoring
- Performance profiling
- Test coverage tools
- Deployment, packaging, versions
- Console logging

Developer Adjustments

- Syntax: ! . , -> _ ;
- Single assignment:
- Recursion rules!
- ( ) vs [ ] vs { } vs << >>
- List maps and folds
- Function overloading with pattern match

```
fact(0) -> 1;
fact(N) when N>0 ->
    N * fact(N-1).
```

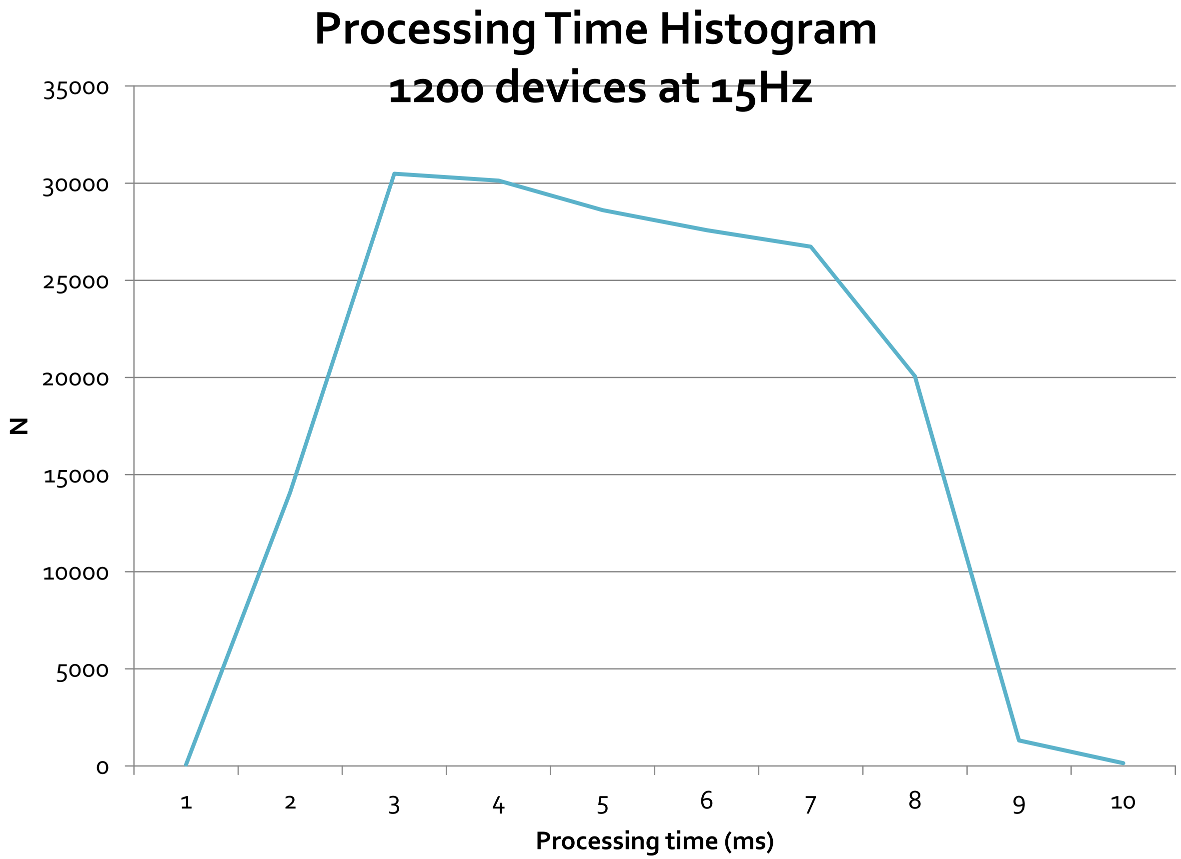
- Use the Erlang tools!
  - Find out what they are
  - How do they work



Erlang loses

- Support for vxWorks and its harder real-time
- But! Do we care?
  - Same groups lamenting loss of vxWorks also moving more processing to FPGA hardware
  - Computers are pretty fast, Erlang is pretty efficient.

Erlang Front-end Processing



D. Nicklaus, ICALEPCS 2011

Accomplishments

- Fully Functional Front End Framework
  - All required protocols (read, set, plot, alarm,…)
- A few test device drivers implemented:
  - Simple cache (settings reflected to readings)
  - Picomotor over TCP
  - Nova Near Detector monitoring
- Erlang-C++ interface available
- Deployed with ACSys-in-a-Box Project
  See: MOPMU039

Why Erlang??

Why not C++?

Technical Reasons

- Concurrency, distributed this
- Functionallist management that
- Reliability, high-availability so-and-so
- Blah blah blah real-time under Linux
- And so on and so forth, et cetera, et cetera

What Happened in 1984?

Better reasons

- Motivation
  - Learn something new.
- Innovation
  - “Functional programming is ‘en vogue’” – M. Voelter, Tuesday morning.
- Productivity
  - Let the Run Time system do the work, not the programmer
- Fun

Erlang Front End Team

- Rich Neswold
- Charlie Briegel
- Jerry Firebaugh
- Jimmy You
- Mike Sliczniak
- Bob Goodwin
- Ron Rechenmacher
- Charlie King

Thank you!