Programming Language Impact on the Development of Distributed Systems

Steve Vinoski
Architect, Basho Technologies
Cambridge, MA USA
vinoski@ieee.org
@stevevinoski
http://steve.vinoski.net/

Co-Authors

- Debasish Ghosh, Anshin Software, Kolkata, India
- Kresten Krab Thorup, Trifork, Aarhus, Denmark
- Justin Sheehy, Basho Technologies, Cambridge, MA USA

TL;DR

- RPC was invented in RFC 707 in 1975, Nelson RPC implementation paper in 1981
- 80s dist. sys. research focused on distribution of objects, often ignoring distributed failure modes
- Systems like CORBA, COM, J2EE, and WS-* focused on
  - language first
  - major languages (Java, C++)
  - distributed systems second

basho

Alternative Languages

- Today the web, the cloud, and “big data” systems often use languages like Ruby, Python, PHP, JavaScript, etc.
- Integration often occurs at the web level, not in traditional middleware
- HTTP finally provides true interface/implementation split
- Allows more freedom of language choice for e.g. time-to-market, maintenance ease

basho

Erlang is our solution to three problems regarding the development of highly concurrent, distributed "soft real-time systems":

- *To be able to develop the software quickly and efficiently*
- *To have systems that are tolerant of software errors and hardware failures*
- *To be able to update the software on the fly, that is, without stopping execution*

— Mike Williams, co-inventor of Erlang
(quote from "Erlang Programming" by F. Cesarini and S. Thompson)

basho

Erlang Origins

- Mid-80's, Ericsson Computer Science Laboratories (CSL)
- Joe Armstrong began investigating languages for programming next-generation telco equipment
- Erlang initially implemented in Prolog with influence and ideas from ML, Ada, Smalltalk, other languages

basho

Erlang History

- Robert Virding and Mike Williams joined Joe to continue Erlang development
- 1991: Claes “Klacke” Wikström added distribution to Erlang
- 1993: first commercial version of Erlang
- 1996: Open Telecom Platform (OTP)
- 1998: Erlang goes open source, available from www.erlang.org
- Growing ever since, especially past 5 years

basho

They Come for the Concurrency...

- What often attracts developers to Erlang is its concurrency support
- Erlang processes are very lightweight and inexpensive
- Concurrent apps easier than in Java, C++, etc. — no dealing with error-prone concurrency primitives

"What if the OOP parts of other languages (Java, C++, Ruby, etc.) had the same behavior as their concurrency support? What if you were limited to only creating 500 objects total for an application because any more would make the app unstable and almost certainly crash it in hard-to-debug ways? What if these objects behaved differently on different platforms?"

— Joe Armstrong, father of Erlang

basho

...But They Stay for the Reliability

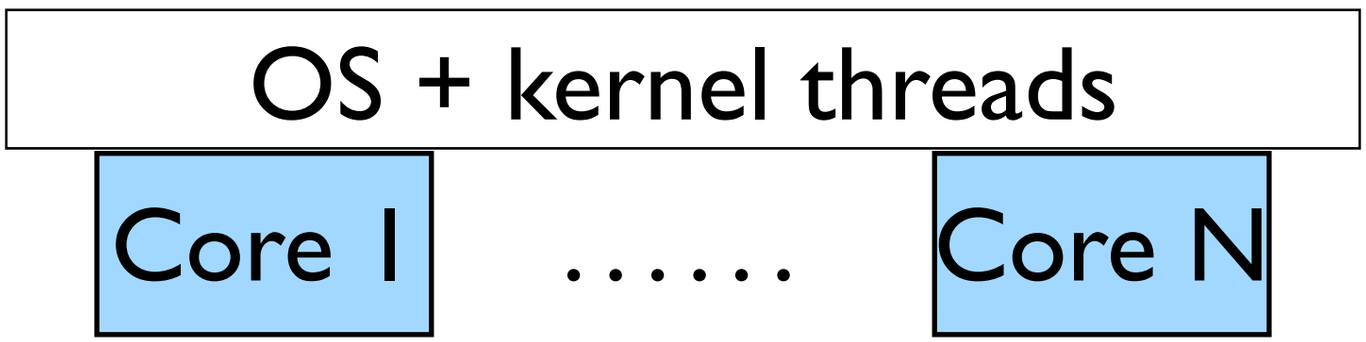
- Erlang's concurrency directly supports its strong reliability
- Shared-nothing approach yields reliability, scalability
- Cheap recovery: if something goes wrong, let it crash, start a new one
- True multiprocessing: start processes on different cores/hosts

basho

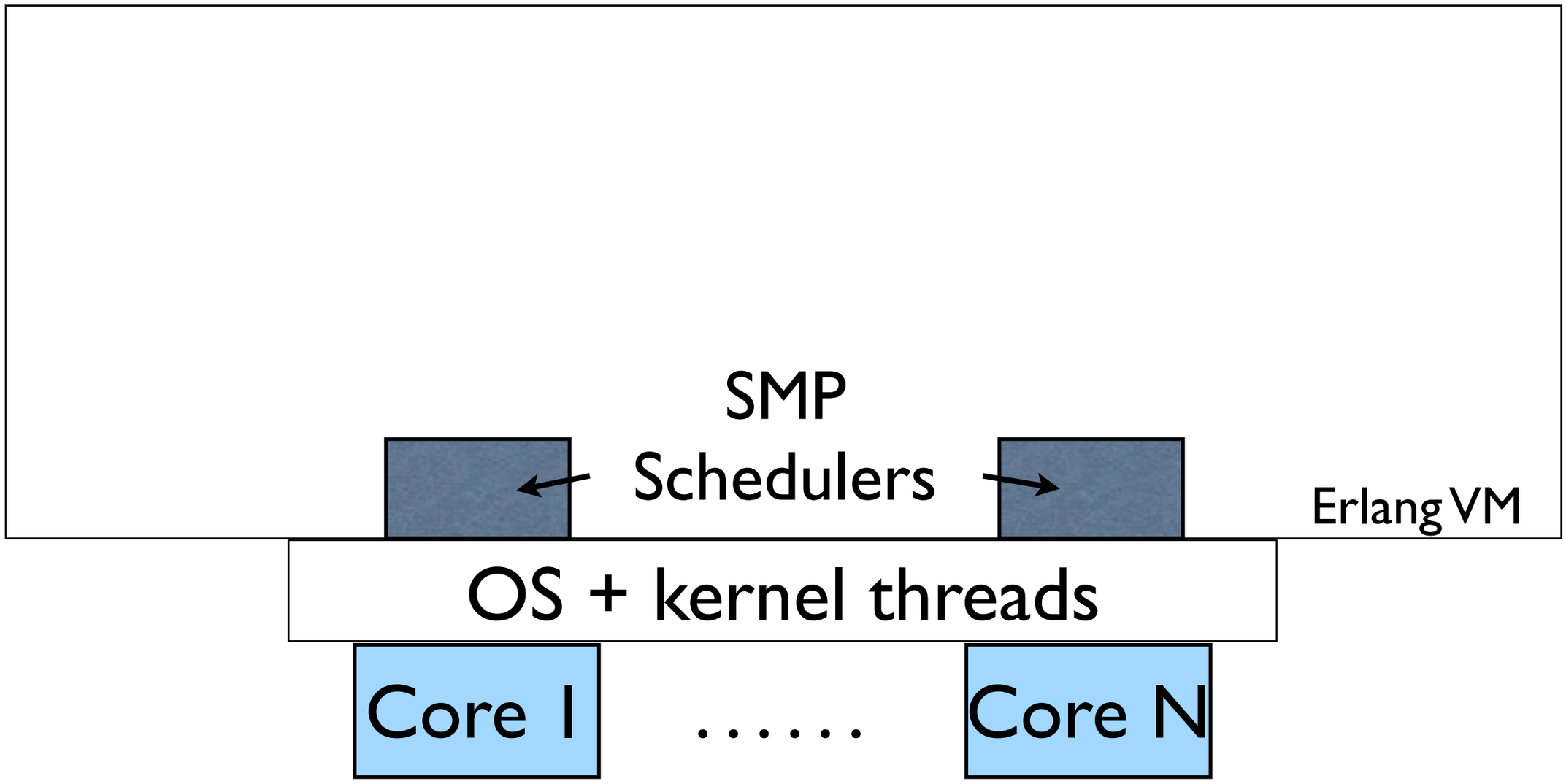
Reliability and Processes

- Isolation: processes communicate only via message passing (no shared state)
- Distribution: need 2+ computers for a reliable system, so process model must work across nodes
- Monitoring and supervision: allow one process to detect when another one fails and take appropriate action

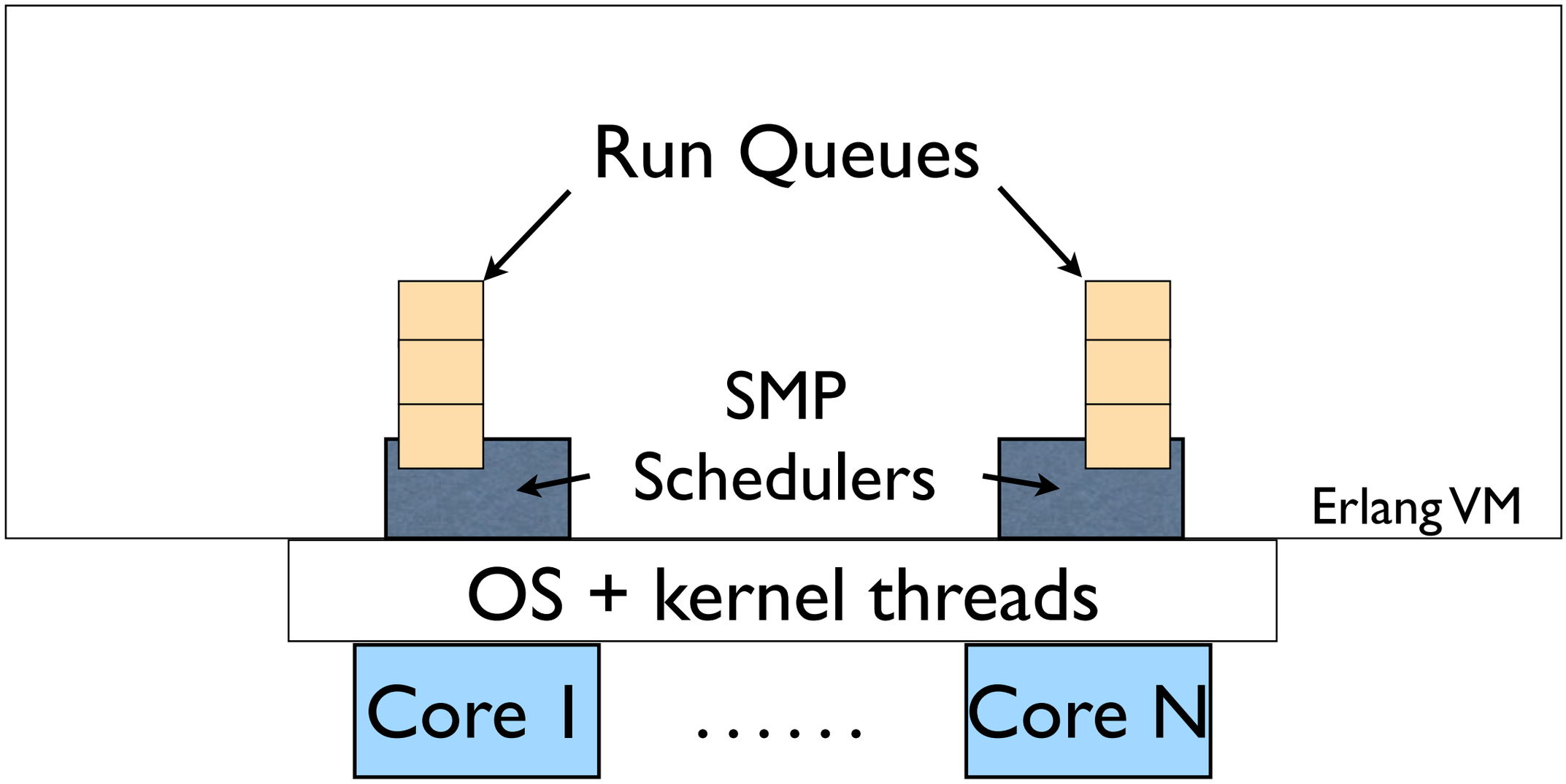
Process Architecture



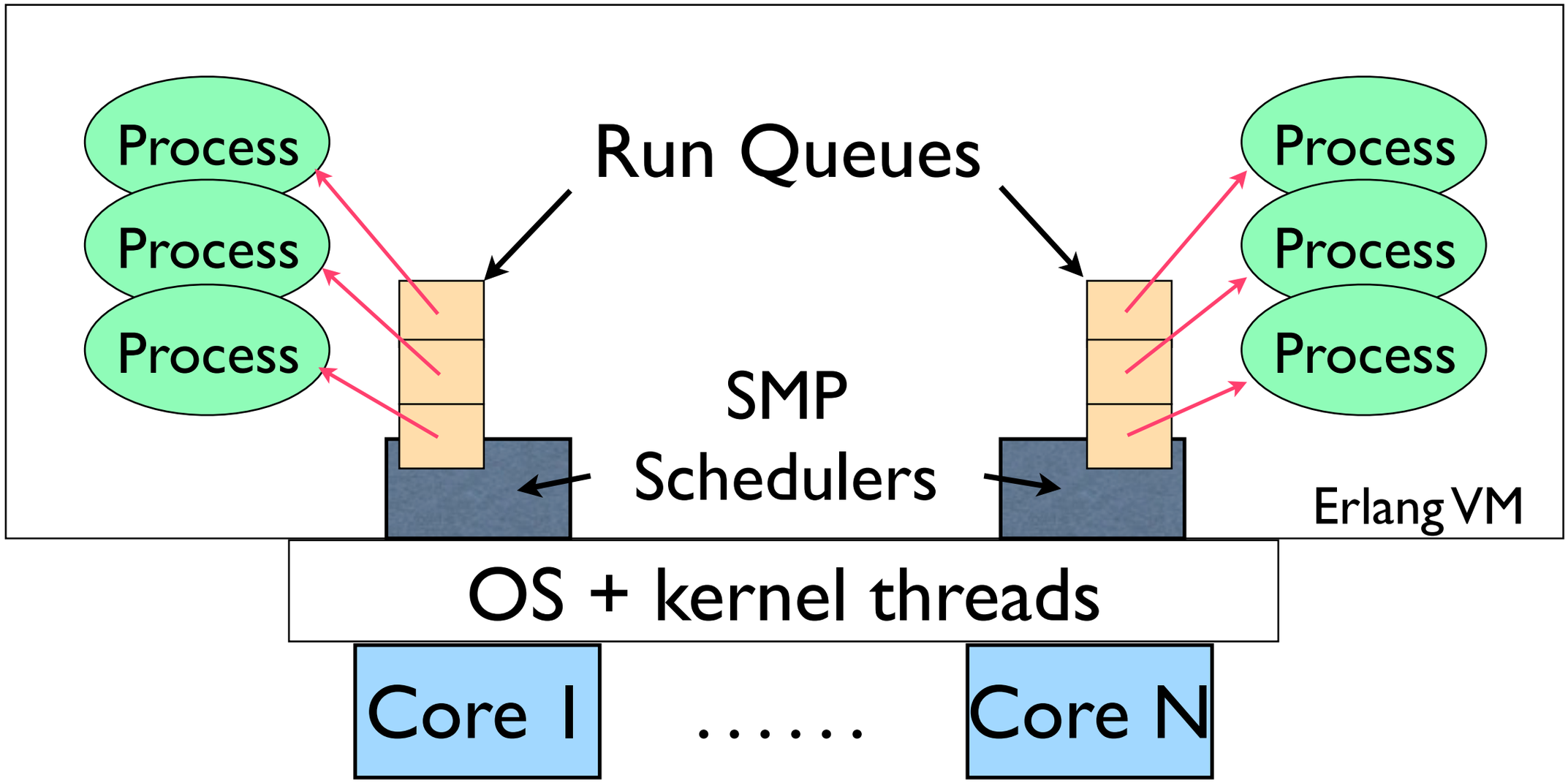
Process Architecture



Process Architecture



Process Architecture



Process Linking

- One process can monitor another by:
  - Linking: bidirectional, each process is tied to the other (erlang:link/1, erlang:spawn_link/1,2,3,4)
  - Monitoring: one process monitors another (erlang:monitor/2)
- Allows processes to take action when another one dies
- And works across multiple nodes

basho

Matching/Binding

- Binding variables via matching can apply to multiple variables at once

{{Year, Month, Day}, {Hour, Min, Sec}} = calendar:local_time().

- Result: {{2011,12,13},{10,32,47}}
  - Year=2011, Month=12, Day=13
    Hour=10, Min=32, Sec=47

basho

Binary Matching

- Matching a TCP header (from Cesarini & Thompson “Erlang Programming”)

```
<<SourcePort:16, DestinationPort:16,
  SequenceNumber:32, AckNumber:32,
  DataOffset:4, _Reserved:4, Flags:8,
 WindowSize:16, Checksum:16,
  UrgentPointer:16, Payload/binary>> = TcpBuf.
```

basho

And So Much More

- Relatively simple language with few constructs
- Capabilities for foreign system integration
- Functional programming (immutable variables, higher-order functions, tail recursion, etc.)
- Standardized OTP frameworks for servers, state machines, event handlers, supervisors, applications
- Choose from many open source libraries and components

basho

What About the JVM?

- Akka: Erlang-inspired middleware in Scala on the JVM
- Erjang: implementation of Erlang on the JVM

Challenge

- Today's web, cloud, "big data" systems still need modularity, failover, redundancy, consistency, availability, etc.
- Many are implemented in alternative languages
- Research challenge: how to take further advantage of alternative languages like Erlang, Akka, Lua, JavaScript, etc. to build better large-scale systems, without a "language first" focus

basho