
Presented by Xiang Zhang
CSE294@UCSD
March 5, 2010
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Summary
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Summary
The work was performed at the Ericsson CSLab during 1981 – 2003.

During this time, Erlang PL and OTP (Open Telecom Platform) was developed to develop fault-tolerant systems.

A number of large applications were written using Erlang/OTP system.
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Summary
A LIST OF REQUIREMENTS FOR TELECOM SYSTEMS

- Concurrency
- Soft real-time
- Distributed
- Hardware interaction
- Large software systems
- Complex functionality
- Continuous operation
- Fault tolerance
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Outline
THE DESIGN PHILOSOPHY

The software is organized into a hierarchy of tasks that the system has to perform.

- Try to perform the top level task.
- If errors occur, try to correct them; if correction fails, try to perform a simpler task.
ERROR ISOLATION

- Errors should not be able to propagate.
- Protection domain needed!
- Use the concept of process to provide protection domain.
PROCESS

Process properties:
- Isolated Memory Space
- No data sharing
- Lightweight
- Message Passing
- Provide an architectural infrastructure
- Portability
CONCURRENCY ORIENTED PROGRAMMING

- The COP view of the world - the world is concurrent

- How to program in this style?
  1. Identify all the truly concurrent activities in our real world activity
  2. Identify message channels between the concurrent activities
  3. Write down all the messages which can flow on message channels
  4. Write the program by following the structure of the problem
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Summary
LANGUAGE SUPPORT

- Erlang PL was invented to support such an architecture.
- Sequential Erlang
- Concurrent programming
OPEN TELECOM PLATFORM

Applications written in other languages

Applications written in Erlang

Mnesia

SNMP
Agent

Web
Server

Erlang Run-time system

Commercial Operating System and computer hardware
OUTLINE

- Background
- Problem Domain
- The architectural model
- Erlang/OTP
- The structure of a fault-tolerant system
- Summary
EXCEPTIONS, ERRORS, FAILURES

Normal State

Exception Handler

corrected

catch

Exception

throw exception

Failure State

no such exception handler

Exception

Generate exception

Internal error

Erlang Virtual Machine

Internal error
WHAT HAS OTP PROVIDED?

- OTP behavior
  1. Process design pattern
  2. Implemented in library modules
  3. Does generic process work and error handling

- OTP behavior includes:
  1. workers
  2. supervisors
SUPERVISION HIERARCHIES

The hierarchy of tasks is implemented as a supervision hierarchy

1. Associate a *supervisor process* to each task
2. The supervisor assigns a *worker* to try to perform the task
3. Supervisors and workers are arranged into hierarchical trees
AN EXAMPLE

Linear supervision

- If my parent stops me then I should stop all my children.
- If any of my children dies, then I must try to restart that child.
ANOTHER EXAMPLE

And/or supervision hierarchy

1. If my parent stops me then I should stop all my children.
2. If any child dies and I am an AND supervisor stop all my children and restart all my children.
3. If any child dies and I am an OR supervisor restart the child that dies.
Many modern distributed systems require concurrency and fault tolerance

Architectural model for a fault-tolerant system

Erlang supports concurrency oriented programming very well

OTP libraries provide behaviors supporting fault tolerant programming

Fault-tolerant system can be organized into a hierarchy of tasks using Erlang/OTP
REFERENCES
