Fault Tolerant Concurrent C: A Tool for Writing Fault Tolerant Distributed Programs

Fault Tolerant Systems Project

Dominik Harmim

xharmi00@stud.fit.vutbr.cz

Brno University of Technology, Faculty of Information Technology

6th March 2020

Cmelik, R.F.; Gehani, N.H.; Roome, W.D.: **Fault Tolerant Concurrent C: A Tool for Writing Fault Tolerant Distributed Programs**. In *The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*. Tokyo, Japan: IEEE. June 1988. ISBN 0818608676. pp. 56–61. ISSN 07313071. doi:10.1109/FTCS.1988.5297.
https://ieeexplore.ieee.org/document/5297

Motivation

- Concurrent C is a superset of C that provides parallel programming facilities.
- The LAN multiprocessor implementation in AT&T led to explore the design of a fault tolerant version of Concurrent C – FT Concurrent C.
- FT Concurrent C can be used to write portable programs that will continue to operate with full functionality despite the failure of some processors.
- To reduce the cost of fault tolerance, the stress is put on selective fault tolerance (only the critical parts of programs are made fault tolerant) by replicating critical processes.

Fault Tolerance Approaches

1. Hardware Approach – uses redundant, fault tolerant hardware. The advantage is that this usually does not require additional programming effort. However, it usually requires specialised hardware.

2. Transparent Approach – the underlying operating system transparently provides fault tolerance. This involves duplicating processes and/or saving state on stable memory. It works on a variety of hardware and there is not required additional programming effort.

3. Tools Approach – provides programming facilities that allow writing programs with fault tolerant critical parts. In exchange for some additional programming effort, the programmer gets efficient fault tolerance. Based on stable storage or replication of program components.

Selected Approach to Fault Tolerance

- It has been chosen the tools approach for FT Concurrent C because it allows selective system fault tolerance to be achieved at a reasonable cost.
- The system designer can best specify the parts of the system that should be fault tolerant.
- FT Concurrent C uses the replicated process model. A replicated process consists of a set of identical processes (replicas) that execute the same algorithm.
- Critical processes are replicated and the replicas are placed on different (identical) processors. The program will continue to operate as long as at least one of these replicas is alive.

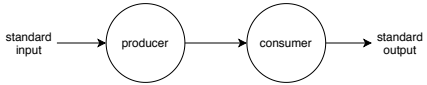
Concurrent C

- A Concurrent C program consists of a set of processes that execute in parallel.
  - Processes are instances of the data type `process`.
  - `process spec/body` are keywords for defining processes.
  - Processes are created using the keyword `create`.

- Two processes interact by first synchronising and then exchanging information using the concept of a transaction.
  - Transactions are defined using the keyword `trans`.

- The process requesting service via a transaction is automatically forced to wait until the requested transaction is processed. The transaction results are then sent back to the waiting client.
  - Processes accept transaction calls using the `accept` statement.

Concurrent C – Example




```
1  process spec consumer() { trans void send(int); };
2  process spec producer(process consumer);
3  process body producer(process consumer cons) {
4      int c; while ((c = getchar()) != EOF) cons.send(c);
5      cons.send(EOF);
6  }
7  process body consumer() {
8      int ch; while (1) {
9          accept send(c) ch = c;
10         if (ch == EOF) break;
11         if (islower(ch)) ch = toupper(ch);
12         putchar(ch);
13     }
14 }
15 int main(void) {
16     create producer(create consumer()); return 0;
17 }
```

- A superset of Concurrent C. Its extensions include facilities for creating replicated processes, detection of failure of processes, automatic termination of slave orphan processes, etc.

- **Assumptions:**
  1. Processor failure can be detected.
  2. The processors are homogeneous and fail by stopping.
  3. Each message is either transmitted correctly or not at all.
  4. All working processors can communicate with each other.

FT Concurrent C – Replication Model

- Replicated process behaves like a single process. Interaction with a replicated process automatically implies interaction with all of its replica processes. All replicas generate replies.
- First-response approach – just the response of the first replica is taken and the responses of the other replicas are discarded.
  - Protects against processor failures.
  - Only one active replica is needed for full functionality.
  - It is fast because a process interacting with a replicated process does not have to wait for all the replicas to respond.
  - Replicas can fall behind in execution.
- **Programmers responsibilities:**
  1. Decide which processes should be replicated.
  2. Ensure that all replicas of a replicated process have the same external behaviour.

Creating a replicated process with *n* copies, each with identical arguments:

```
create [slave] process_type(arguments)
  [copies(n) | processor(n1, n2, ...)]
```

- The operation returns a single process identifier which identifies the entire set of replicas.
- The created process can be marked as a slave of the parent process. If the parent process terminates abnormally, then all of its slave processes will be killed.

Summary

- FT Concurrent C is a tool for writing fault tolerant distributed programs.
- Critical program components (processes) can be made fault tolerant by replicating them.
- Interaction with the replicated processes is managed by the run-time system.
- The replication is not for free. A price has to be paid because of the:
  1. extra transaction calls,
  2. synchronisation between replicas,
  3. programming effort.