Concurrent Programming Languages

Alexis Beingessner Troy Hildebrandt

March 2, 2015

1 Abstract

Developing applications for distributed systems provides unique challenges that many modern programming languages are not inherently equipped for. These challenges are often ones that arise with parallel and concurrent programming, with additional concerns such as node and network failures, software survivability, and network latency issues, among others.

Certain languages attempt to tackle some of the major problems associated with developing distributed applications. These languages provide built-in facilities for creating nodes that communicate over a network, error handling, message passing, and mechanisms for more transparently simulating shared memory. This simplifies fault-tolerance and synchronous access to data across multiple machines.

We look at several languages and their respective extensions that enable distributed application development, including Erlang, Concurrent/Cloud Haskell, and Ada. For each, we examine their strengths and innovations, and assess their impact on distributed computing. The languages explored all provide unique solutions to some of distributed programming's toughest problems, in some cases using completely different programming paradigms.

In addition, we look at how these techniques can be used to increase the security and reliability of applications that are only running locally. For instance, separating an application into several logical processes allows those processes to fail or become compromised, without affecting other parts of the application.

2 Argument

- Introduction: What is concurrent programming; what challenges does it involve; why solving the problem at the system level (as opposed to application level) is wrong; problems with using C-like languages for it.
- Ada
- Erlang
- Cloud Haskell
- Using techniques locally
- Conclusion

[9][10][6][1][3][4][7][8][2][5]

References

[1] B. Anderson, L. Bergstrom, D. Herman, J. Matthews, K. McAllister, J. Moffitt, S. Sapin, and M. Goregaoka. Servo experience report.

[2] J. Armstrong. Concurrency oriented programming in erlang. *Invited talk, FFG*, 2003.

[3] J. Armstrong. *Making reliable distributed systems in the presence of software errors*. PhD thesis, The Royal Institute of Technology Stockholm, Sweden, 2003.

[4] D. Cornhill. A survivable distributed computing system for embedded application programs written in ada. *ACM SIGAda Ada Letters*, 3(3):79–87, 1983.

[5] C. B. Earle, L.-Å. Fredlund, and J. Derrick. Verifying fault-tolerant erlang programs. In *Proceedings of the 2005 ACM SIGPLAN workshop on Erlang*, pages 26–34. ACM, 2005.

[6] J. Epstein, A. P. Black, and S. Peyton-Jones. Towards haskell in the cloud. In *ACM SIGPLAN Notices*, volume 46, pages 118–129. ACM, 2011.

[7] N. H. Gehani and T. A. Cargill. Concurrent programming in the ada® language: The polling bias. *Software: Practice and Experience*, 14(5):413–427, 1984.

[8] T. Jim, J. G. Morrisett, D. Grossman, M. W. Hicks, J. Cheney, and Y. Wang. Cyclone: A safe dialect of c. In *USENIX Annual Technical Conference, General Track*, pages 275–288, 2002.

[9] P. Rogers and A. J. Wellings. The application of compile-time reflection to software fault tolerance using ada 95. In *Reliable Software Technology–Ada-Europe 2005*, pages 236–247. Springer, 2005.

[10] U. Wiger and E. T. Ab. Four-fold increase in productivity and quality-industrial-strength functional programming in telecom-class products. 2001.