Reliability and Manycores – Challenges and Benefits

Björn Döbel, TU Dresden OS Group

September 9, 2013

In the near future, hardware vendors will be able to supply us with processors containing hundreds or even thousands of cores. These cores will be built from very small transistors, which are in turn more vulnerable to failures resulting from cosmic radiation, thermal, and aging effects. While currently, these errors are still handled mostly using hardware-level fault tolerance mechanisms, these techniques will be insufficient or too expensive to deal with the error levels of future systems. Therefore, software-level fault tolerance methods will become more important.

In this talk I am going to introduce one such software-level solution: The L4/Fiasco.OC microkernel comes with *Romain*, a user-level service that allows to transparently replicate unmodified binary-only applications. I will explain how Romain manages replicas and their resources to achieve low runtime overheads for replicated execution (geometric mean: 2.51% for triple-replicated execution of the SPEC INT 2006 benchmarks). I will furthermore discuss how we replicate multithreaded applications, where the inherent scheduling-induced nondeterminism makes traditional state-machine replication inefficient.

During this research project we also found that depending on the actual workload, placing replicas on different CPU sockets may benefit performance (for replicas with large cache footprints) whereas it may hurt other scenarios that heavily rely on fast inter-processor messages. I will highlight these problems, which will become even more relevant in large-scale manycores and argue that the replication infrastructure needs to thoroughly monitor application behavior to optimize replica placement.

With Romain protecting user-level software on top of L4/Fiasco.OC, we are left with a small set of components that are required to never suffer from hardware errors in order for replication to work correctly. We call these components the *Reliable Computing Base (RCB)*, which for our system includes the OS kernel itself as well as the replication infrastructure. I will explain that all software fault tolerance techniques possess an RCB, although it varies across solutions. Finally, I will discuss potential approaches for protecting RCB code against hardware errors in order to get a completely protected system.