Contribution ID: **52** Type: **Poster**

Analysis of DIRAC's behavior using model checking with process algebra

Thursday 24 May 2012 13:30 (4h 45m)

DIRAC is the Grid solution designed to support LHCb production activities as well as user data analysis. Based on a service-oriented architecture, DIRAC consists of many cooperating distributed services and agents delivering the workload to the Grid resources. Services accept requests from agents and running jobs, while agents run as light-weight components, fulfilling specific goals. Services maintain database back-ends to store dynamic state information of entities such as jobs, queues, staging requests, etc. Agents use polling to check for changes in the service states, and react to these accordingly. A characteristic of DIRAC's architecture is the relatively low complexity in the logic of each agent; the main source of complexity lies in their cooperation. These agents run concurrently, and communicate using the services' databases as a shared memory for synchronizing the state transitions.

Although much effort is invested in making DIRAC reliable, entities occasionally get into inconsistent states, leading to a potential loss of efficiency in both resource usage and manpower. Tracing and fixing the root of such encountered behaviors becomes a formidable task due to the inherent parallelism present. In this paper we propose the use of rigorous methods for improving software quality. Model checking is one such technique for analysis of an abstract model of a system , and verification of certain properties of interest. Unlike conventional testing, it allows full control over the execution of parallel processes and also supports exhaustive state-space exploration.

We used the mCRL2 language and toolset to model the behavior of two critical and related DIRAC subsystems:the workload management and the storage management system. mCRL2 is based on process algebra, and is able to deal with generic data types as well as user-defined functions for data transformation. This makes it particulary suitable for modeling the data manipulations made by DIRAC's agents. By visualizing the state space and replaying scenarios with the toolkit's simulator, we have detected critical race-conditions and livelocks in these systems, which we have confirmed to occur in the real system. We further formalized and verified several properties that were considered relevant. Our future direction is exploring to what extent a (pseudo)automatic extraction of a formal model from DIRAC's implementation is feasible. Given the highly dynamic features of the implementation platform (Python), this is a challenging task.

Student? Enter 'yes'. See http://goo.gl/MVv53

Yes

Author: REMENSKA, Daniela (NIKHEF (NL))

Co-authors: Prof. BAL, Henri (Professor of Computer Science); TEMPLON, Jeff (NIKHEF (NL)); Dr VERSTOEP, Kees (Scientific Programmer); Prof. WILLEMSE, Tim (Assistant Professor)

Presenter: REMENSKA, Daniela (NIKHEF (NL))

Session Classification: Poster Session

Track Classification: Software Engineering, Data Stores and Databases (track 5)