BMC Software Inc.

Technical Disclosure Publication Document

CMDBf Push Pull Mode Hybridization

Author

Vincent J. Kowalski

Posted: June 2011

Overview

This document describes an invention referred to as CMDBf Push/Pull Mode Hybridization. By creating an architecture in which both Push Mode and Pull Mode federation are present, this invention avoids a number of the pitfalls encountered by relying solely on one of these approaches while enjoying most of the benefits. By avoiding an approach that pits Push Mode versus Pull Mode and embracing both, an architecture for federation that is both performant and real-time is achieved.

Background

The CMDBf standard is now being adopted across the industry. The standard mandates that to be compliant an implementation must be based on Push Mode or Pull Mode federation (or both). Each of these federation modes has inherent advantages and disadvantages. This invention takes a “best of both worlds” approach to CMDB Federation by incorporating the advantages of each mode while minimizing the disadvantages. To accomplish this, two architectural scenarios are presented, each of which is a hybrid of both Push and Pull Mode CMDB federation.

Solution: CMDBf Push/Pull Mode Hybridization

As stated previously, this invention is the hybridization between Push and Pull Mode federation with the intent of capitalizing on the advantages of both while minimizing their drawbacks. There are two distinct forms this hybridization can take:

- Primary pull
- Primary push

Each of these has an associated architecture, which is explained in the subsections that follow.

Primary Pull Architectural Scenario

In the Primary Pull form of the hybridization, we start with a Pull Mode federation architecture. Iinstead of merely resolving all queries dynamically and on-demand, however, we introduce an embedded Management Data Repository ("MDR") similar to that used in traditional Pull Mode federation. Remember, this embedded MDR is required by Push Mode federation because the results of invoking the registration service (i.e., the pushed instances of data from the MDRs) must be stored somewhere. Instead of the embedded MDR being populated by invoking the registration service, however, it is populated by queries that are performed by the federating CMDB. Figure 1 below is a depiction of the Primary Pull architectural scenario. The advantage of this approach is that data that is retrieved anyway will get stored in the embedded MDR. On subsequent query executions, the data in this embedded MDR can be used instead of having to always delegate the queries down to the MDRs. Obviously, at system initiation, there will be no advantage to this architectural scenario since the embedded MDR will be empty, but over time, as the content of the embedded MDR grows, queries should become more performant, and scalability issues with the traditional Pull Mode architecture will be avoided or, at least, minimized. As such, the holding area (i.e., the embedded MDR) that is normally used in Push Mode federation is used to optimize this Primary Pull architectural scenario.

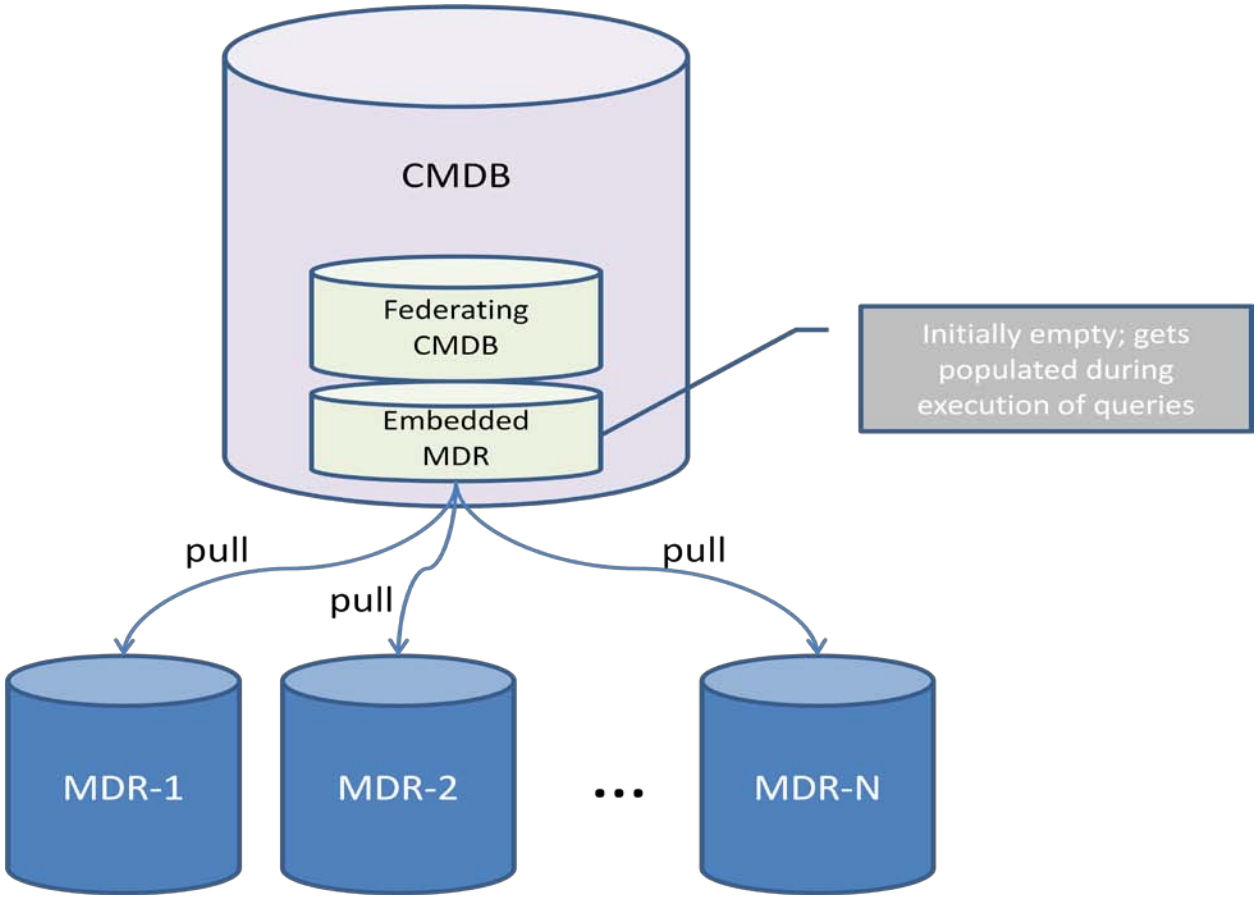

Figure 1: Primary Pull Architectural Scenario

Primary Push Architectural Scenario

In the Primary Push form of the hybridization, we start with a Push Mode federation architecture. As such, there is an initial population of the embedded MDR that is performed by the CMDBf registration service, but instead of doing a re-population of the embedded MDR at some pre-defined intervals, there are triggers set up to push data in an event-driven manner. These triggers work as follows: when there have been changes to the MDR (in database terms: insert, update, and delete operations), the trigger is fired. When this trigger is fired, changes (e.g., insert, update, delete) to the contents of an underlying MDR causes a Δ registration operation to be invoked. (The Primary Push architectural scenario is depicted in Figure 2 below.) In other words, only the data that is changed is pushed to the embedded MDR, keeping the embedded MDR up-to-date and avoiding large, costly bulk push operations. As such we use the on-demand nature of the Pull Mode in the Primary Push architectural scenario.

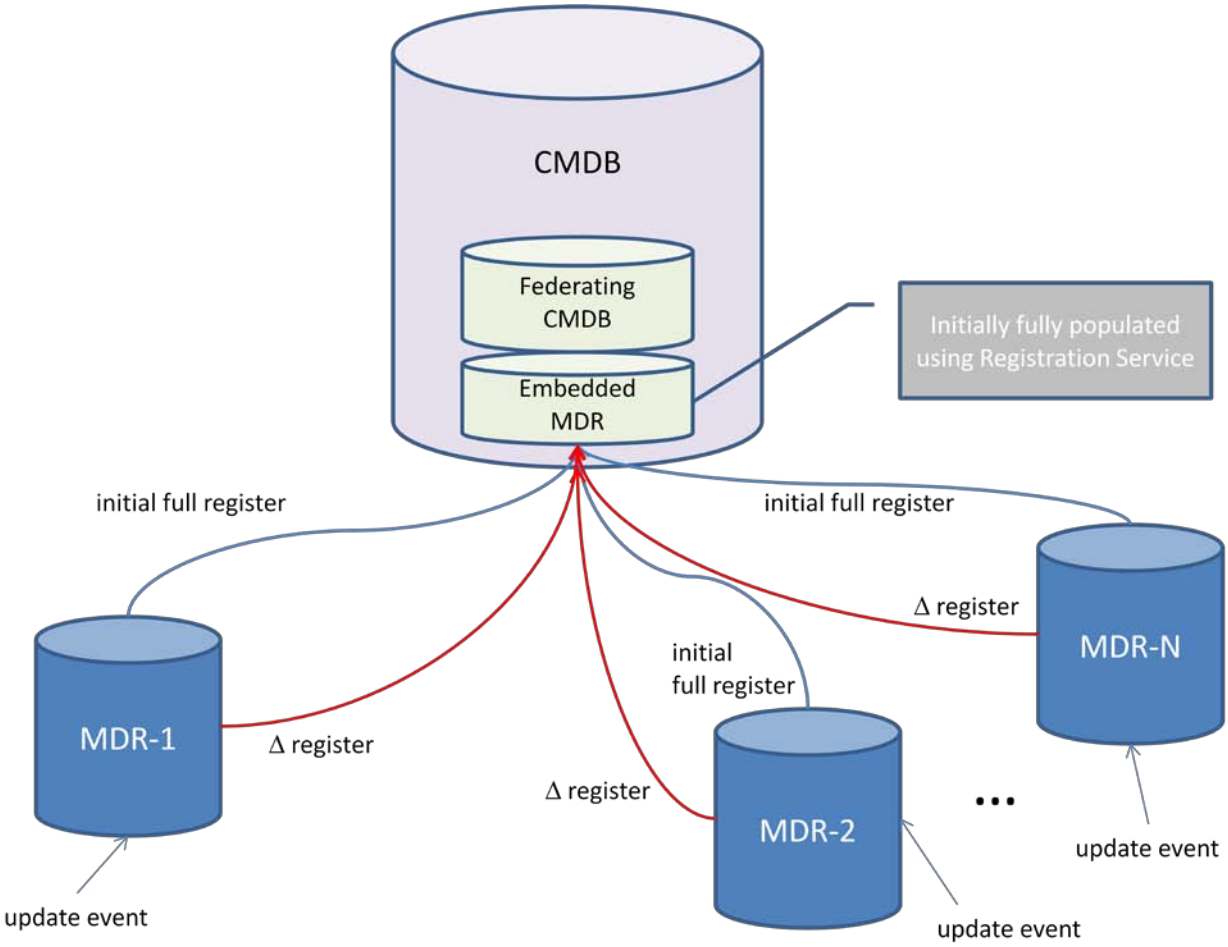

Figure 2: Primary Push Architectural Scenario