This document summarizes ADDB discussions in the C2 architecture team. The goal is to provide a starting point for design level discussions and resource estimation. Note that this documents is not a substitute for a list of requirements or use-cases. Intended audience includes T1 team and C2 architects.

Overview.

Analysis and Diagnostics Data-Base (ADDB) contains information describing ongoing activity of C2 system. This information is stored on persistent storage and can be later queried and retrieved to analyze system behavior. See ADDB View Packet and ADDB QAS list for more information.

Commented [1]: what's the purpose of such DB and who will be the primary user(s) of it? [I did read the entire doc]

Commented [2]: I sent Kevin the link to this as a very brief "overview" of what ADDB encompassed, as stated at the top, use cases are out of scope of this document. Some of the other ADDB documents located on the Seagate Google drive should help answer any specific questions.

Commented [3]: For ease of search: https://drive.google.com/drive/search?q=addb

Items.

- The overall idea of ADDB is to accumulate and store certain auxiliary information during execution of C1 file operations. This information can be later used to analyze past system behavior. Compare this with Lustre logging system. The critical differences are:
  - ADDB has format suitable for later processing, including querying;
  - ADDB is populated systematically rather than in *ad hoc* manner;
  - ADDB is designed to be stored on persistent storage.

  Note that ADDB is much more than just a glorified logging, see below.
- ADDB consists of records and each record consists of *data points*. A data point is a result of an individual measurement of a certain system parameter or a description of an event of interest.

- Examples of measurement data points are:
  - a particular RPC staid N microsecond in a server queue before its processing started;
  - a client data cache hit ratio is α;
  - a particular persistent storage meta-data read request took M microseconds;
  - a particular meta-data RPC found directory entry in cache.

- Examples of event data points are:
  - memory allocation failed;
  - RPC timed out;
  - disk transfer took longer than a threshold time.

- From the examples above it is clear that measurements fall into two classes: measurements pertaining to a particular file system activity (RPC execution, cache write-back, *etc.*) and measurements not related to any specific activity (resource utilization measurements, queue length, *etc.*).

- ADDB records are written to persistent storage by C2 back-end. There is an important difference between ADDB and other persistent data structures such as FOL, file data or meta-data: ADDB is a useful convenience, but it is not required for system correctness. While undesired, ADDB record loss won't render system inconsistent. As a result, it is not necessary to write ADDB transactionally. ADDB IO can be piggy-backed to other transfers or can be directed to a separate device with a lower throughput.

- ADDB is accumulated during normal system activity. Once written, ADDB record stays on storage for a long time, until ADDB is explicitly pruned by administrator.

- ADDB records should have self-identifiable structure: it should be possible to discover the data-points contained in the record. This is required to make addition of new data-point collecting calls as easy as possible (*i.e.*, no changes to user-level tools) and to make ADDB and ADDB tools interoperable between versions.

- ADDB must contain enough information to recover important aspects of system behavior. Specifically, this means that ADDB should at least contain information about all RPCs executed by the system. This implies some degree of duplication between FOL and ADDB. It is possible to link corresponding ADDB and FOL records to each other, but this introduces a dependency between FOL and ADDB pruning.

- The simplest ADDB use is to *replay* it to see how system behaved in the past. A typical example, is replaying ADDB on the next day to understand why last night MPI job ran slow.

- A more interesting usage is to filter ADDB to concentrate on particular aspects of system, e.g., on requests sent from a particular set of clients or operations against a particular set of files. This is achieved by loading ADDB into a data-base and running queries on it through management tools interfaces.

- Even more interesting ADDB usage is to see how system *would* have behaved if some parameters were different. For example, one can take a ADDB trace of N clients working on a particular set of files and process it to represent a trace coming out of of M clients. Similarly, one can change layout parameters (striping, *etc.*). The resulting ADDB is then fed as an input to a simulator.